uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales
2017/2018

Trabajo Fin de Grado

Estudio de gestores de referencias bibliográficas y conectividad web con la plataforma Zotero

Víctor Díaz Obregón

Tutor
Juan Carlos González Víctores
18 de Octubre, 2018, Leganés

Agradecimientos

Quisiera agradecer a varias personas y organizaciones la ayuda que me han prestado no solo en la realización de este trabajo de fin de grado, sino también en los cuatro años del Grado en Ingeniería Industrial.

Entre ellas, y en primer lugar, mi tutor en este trabajo, Juan G. Victores. Su orientación, paciencia y consejos han hecho posibles la realización de este proyecto.

En segundo lugar, debo agradecer a la Universidad Carlos III de Madrid, sus profesores y a la asociación de robótica. Que me han proporcionado la motivación para finalizar mi grado y los conocimientos clave que me ayudarán en mi futuro laboral.

Por último, desearía agradecer a mi familia su apoyo moral y económico durante todos estos años. Ellos me ha permitido completar mis estudios sin pedir nada a cambio. También a mis amigos del instituto, universidad y compañeros de Erasmus, por su ayuda y las experiencias compartidas.

Resumen

El contenido de este trabajo trata sobre el estudio de gestores de referencias bibliográficas y las tecnologías web que permiten la interacción entre dichos gestores y aplicaciones.

El proyecto del que forma parte este trabajo consiste en la creación de una plataforma desde la cual un usuario pueda administrar referencias a documentos científicos en diferentes gestores de referencias bibliográficas al mismo tiempo. La finalidad última es ahorrar tiempo a los investigadores a la hora de gestionar publicaciones. El objetivo del presente trabajo de fin de grado es estudiar la posibilidad de la inclusión del gestor de referencias bibliográficas Zotero en el proyecto final.

Para alcanzar dicha meta, en esta memoria se analizarán los siguientes conceptos. Primero, serán tratados los aspectos legales (protección de datos) y el entorno socioeconómico del trabajo. Segundo, se introducirán los conceptos de gestores de referencias bibliográficas y tecnologías web. Finalmente, se procederá a la implementación de los conceptos introducidos proyecto y se comentarán los resultados obtenidos.

Abstract

The content of this work deals with the study of bibliographic references managers and the web technologies that allow the interaction between these managers and applications.

The project this work belongs to, consists in the creation of a platform from which a user can manage references to scientific documents in different bibliographic references managers at the same time. The ultimate goal is to save researchers time when managing publications. The objective of the present thesis is to study the possibility of the inclusion of the bibliographic reference manager Zotero in the final project.

To achieve this goal, the following concepts are to be discussed in this report. First, the legal treaties and the socio-economic environment will be studied. Second, the concepts of bibliographic references and web technologies managers will be analyzed. Finally, an implementation of these concepts will be carried out and the results obtained will be commented on.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Este capítulo del trabajo está dedicado a introducir el problema que se quiere solucionar, las motivaciones que llevaron a la realización de este trabajo, plantear posibles soluciones y la contribución del trabajo al mundo científico. Se incluye además un apartado dedicado a la organización del trabajo realizado, que cuenta con un diagrama Gantt y finalmente, una sección en la que se introducen los capítulos del documento con un pequeño resumen del contenido de cada uno.

Nótese que a lo largo del documento se hace referencia a los términos trabajo y proyecto. La palabra trabajo se refiere al trabajo de fin de grado que se esta llevando a cabo, y proyecto hace referencia al objetivo último; la creación de un gestor de referencias multiplataforma.

1.1. Situación del proyecto

En esta sección se introducirá la motivación que llevó a la realización de este trabajo, así como su justificación y su contribución al mundo científico.

1.1.1. Motivación

Con el objetivo de que la investigación de un científico tenga la mayor visibilidad e impacto posible, es fundamental que sea leída, revisada y contrastada por el mayor número miembros de la comunidad científica posible. Existen redes sociales dedicadas a la investigación, con servicios de gestión de referencias bibliográficas que ayudan a conseguir estos objetivos.

La interacción con dichas redes no es siempre sencilla, existen varios problemas inherentes con su uso. El primer inconveniente para un investigador que quiere gestionar su trabajo en estas redes es que debe poseer cuentas separadas en cada plataforma; esto conlleva mantener un un registro de nombres de usuario y contraseñas. El segundo problema consiste en que cada red posee una interfaz de usuario diferente y una manera de organizar el contenido subido distinta. Por último, el método para subir documentos a las distintas redes varía de una plataforma a otra. En la figura 1.1, se pueden observar las diferencias en cuanto a la interfaz entre los gestores de referencias ORCID y ResearchGate. Ambos ofrecen a sus usuarios servicios similares, pero cada uno los presenta de manera distinta.

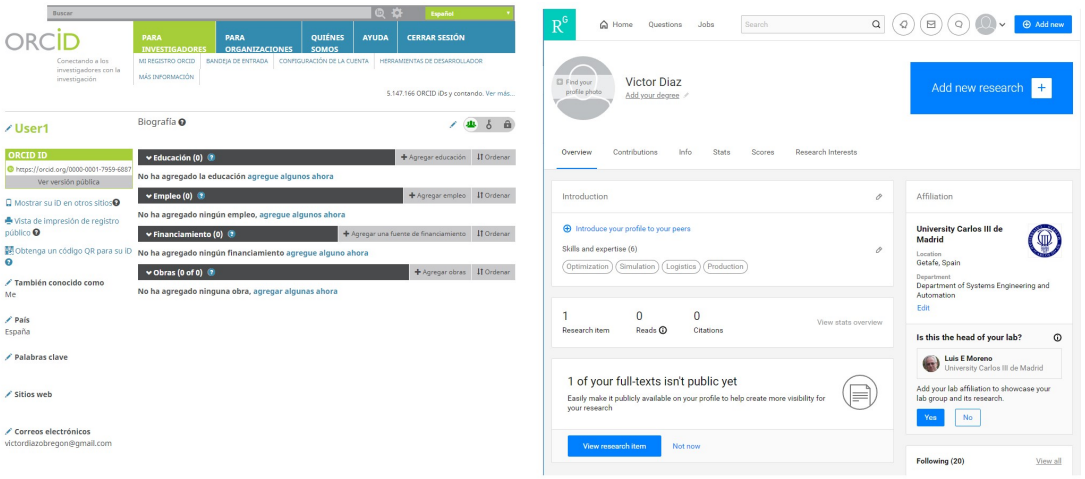

Figura 1.1: Diferencias entre la interfaz de usuario de ORCID y ResearchGate

La motivación de este proyecto surgió de la idea de simplificar la experiencia de interacción con redes sociales para investigadores, en su faceta de gestores de referencias bibliográficas. El concepto es muy similar a una aplicación cuyo uso sea retocar fotos desde un teléfono móvil y además permita que se compartida directamente en distintas plataformas, como pueden ser redes sociales (Facebook, Instagram) o un servicio de mensajera instantánea (e-mail, WhatsApp). Con un simple clic la foto es compartida en las opciones elegidas.

La idea del proyecto de investigación es trasladar este concepto a la gestión de referencias en redes sociales para investigadores. El objetivo es crear una plataforma desde la cual un investigador pueda gestionar referencias a publicaciones, artículos o documentos científicos en varios gestores bibliográficos de Internet, de manera sencilla y eficiente.

1.1.2. Justificación

Existen varias alternativas en la gestión de referencias bibliográficas en redes sociales. Plataformas como Mendeley y Zotero ofrecen una extensión que es posible instalar en navegadores web que permite a sus usuarios generar referencias de contenido online. Las referencias generadas son incluidas automáticamente en la biblioteca de la cuenta del usuario. Otro método con el que obtener referencias, es rellenar los campos en una plantilla proporcionada por el gestor. Por otro lado el usuario puede subir un archivo de extensión .bib generado por él mismo manualmente.

Con este proyecto se pretende solucionar los problemas a los que un investigador se enfrenta al administrar su trabajo entre varias redes sociales, concretamente en la gestión de referencias bibliográficas. El objetivo final consiste en crear una aplicación con la siguiente funcionalidad. En primer lugar, la aplicación dispondrá de una interfaz de usuario simple, que permita elegir archivos que contengan referencias bibliográficas desde el ordenador del usuario. En segundo lugar, será capaz de analizar el archivo y extraer información de este, como autores y título de la publicación. Por último, permitirá al usuario elegir la red social para investigadores a la que se quiere subir la referencia. El documento será subido a la plataforma sin intervención.

1.1.3. Contribución

La contribución de este proyecto al mundo científico, es la automatización del proceso de gestión de referencias bibliográficas en varias plataformas de gestión de referencias, con el objetivo ahorrar tiempo a los investigadores que deseen utilizarla. La tecnología usada en este trabajo, la API (Application Programming Interface o Interfaz de Programación de Aplicaciones) de Zotero, ya está desarrollada y documentada en vídeos, artículos y su página web. Este trabajo se centra en la implementación de dicha tecnología web, con el objetivo de interactuar con la API. Dicha interacción consiste en adquirir permisos de escritura en la biblioteca de Zotero y subir un archivo de referencia bibliográfica. Para ello se utilizará un código escrito en el lenguaje de programación Python.

1.2. Alternativas de diseño

En este apartado se hablará acerca de las distintas iteraciones del trabajo y las partes del diseño que pudieron ejecutarse de manera distinta, antes de optar por la conectividad con Zotero.

Al comienzo del trabajo, el objetivo de este era la automatización de la gestión de referencias bibliográficas en la plataforma ResearchGate. Durante el desarrollo de esta esta etapa, se descubrieron factores que impedían o complicaban en exceso este fin y por ello se encontraban fuera del alcance del autor de este documento. El obstáculo principal encontrado fue que ResearchGate no ofrece a sus usuarios una API pública que permita la interacción con servicios externos. A fin de lograr el objetivo del trabajo, se consideró automatizar el proceso que realizaría un usuario a la hora de subir una referencias, controlando el navegador con un software externo (código de Python). Este método no es fiable ya que si alguna parte de este código fuente HTML de la página cambiará a lo largo del tiempo, el proceso cesará de ser útil.

Por los problemas descritos en el párrafo anterior, se optó por cambiar de plataforma. El nuevo gestor de referencias elegido fue ORCID. ORCID cuenta con una API pública, disponible para todos sus usuarios. El problema surgió al descubrir que dicha API pública únicamente dispone de dos opciones: autenticación de un usuario y lectura de contenido público de usuarios de ORCID. Existe una API *Premium* reservada a usuarios miembros de asociaciones que colaboran con ORCID. Se intentó obtener acceso a dicha API pero fue denegado. Dado que en el futuro proyecto es imprescindible disponer de una herramienta que permita subir contenido a una biblioteca online, no se pudo continuar con el desarrollo del trabajo en ORCID.

Después de dos intentos fallidos, el trabajo continuó con la red social Linkedin. Esta plataforma dispone de una API pública con todas las funciones necesarias para completar este trabajo. Estas herramientas están disponibles a todos los usuario miembros de Linkedin. La implementación de las funciones de la API requiere autorización. A fin de que un usuario pueda convertirse en un desarrollador autorizado debe completar un formulario disponible en la siguiente URL "https://business.Linkedin.com/marketing-solutions/marketingpartners/becomeapartner/marketingdeveloperprogram#getstarted" y esperar a ser validado por el personal encargado de Linkedin. Esta autorización puede tardar hasta tres meses en concederse. Durante el desarrollo del trabajo se solicitó autorización de acceso a los recursos de la API pero nunca fue concedida.

Finalmente, se trató de llevar a cabo el trabajo en la plataforma Zotero, en la cual los objetivos de este trabajo fueron completados. No obstante existen dos alternativas en el diseño que se describe a continuación. El acceso a recursos protegidos en Zotero depende de la obtención de una llave o *API key*. Se distinguen dos métodos con los que conseguir esta *API key*:

1. Método manual

   El usuario es el encargado de conseguir la *API key*, este método se describe en detalle en el apartado 4.2.

2. *OAuth 1.0*

   La obtención de la *API key* está automatizada; utilizando el flujo de autenticación OAuth 1.0. Este método se explica en profundidad en el apartado 3.2.2.

Con la finalidad de alcanzar los objetivos propuestos, en este trabajo se utilizará el método nombrado en la opción número uno debido a su sencillez a la hora de implementarlo.

1.3. Planificación de trabajo

En esta sección se hablará sobre la planificación y organización del presente trabajo de fin de grado. A continuación, se mencionan las distintas fases de dicho trabajo:

- Estudio de gestores de referencias y tecnologías web.
- Implementación de gestores de referencias y tecnologías web.
- Redacción de la memoria del trabajo de fin de grado.
- Revisión de la memoria del trabajo de fin de grado por el tutor.

A continuación se mostrará una diagrama de Gantt (tabla 1.1) que refleja el progreso de cada uno de los objetivos en todas las iteraciones del trabajo.

Tabla 1.1: Diagrama de Gantt del desarrollo del trabajo

Num.	ACTIVIDADES	Desarrollo temporal del trabajo de fin de grado															
		Junio				Julio				Agosto				Septiembre			
		Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4
1	Estudio: ResearchGate																
1.1	Investigación																
1.2	Implementación																
2	Estudio: ORCID																
2.1	Investigación																
2.2	Implementación																
3	Estudio: LinkedIn																
3.1	Investigación																
3.2	Implementación																
4	Estudio: Zotero																
4.1	Investigación																
4.2	Implementación																
5	Redacción y recivisión																
5.1	Investigación																
5.2	Implementación																

1.4. Estructura del documento

En este apartado se procederá a describir brevemente el contenido de cada capítulo que compone este memoria.

- Capítulo primero: Introducción

  Este capítulo está dedicado a introducir el problema que se quiere solucionar, las motivaciones que llevaron a la realización de este trabajo, plantear posibles soluciones y comentar la contribución del trabajo al mundo científico.

- Capítulo segundo: Estudio socio-económico y legal

  Este capítulo se centra en estudiar la percepción en el entorno socio-económico de las tecnologías web y los gestores de referencias bibliográficas, así como el impacto del propio trabajo de fin de grado en este ámbito. También trata las leyes de protección de datos personales y los organismos que regulan dicha legislación. Finalmente, se explica el cálculo del presupuesto de este trabajo.

- Capítulo tercero: Estado del arte

  En este capítulo se introducen los conceptos en los que se fundamenta la investigación realizada en este trabajo de fin de grado. Se estudian los gestores de referencias bibliográficas incluidos en redes sociales para investigadores. Con dicha finalidad se analizan cuatro redes sociales, ResearchGate, ORCID, Linkedin y Zotero. Asimismo, incluye un apartado dedicado al estudio de tecnologías web, donde se definirán los conceptos de API, flujos de autorización OAuth y respuestas de un servidor durante la comunicación.

- Capítulo cuarto: Desarrollo técnico

  En este capítulo se ponen en práctica los conceptos definidos en el estado del arte, con el fin de interactuar con la plataforma Zotero. Se describe el proceso de implementación que permite subir referencias bibliográficas a la biblioteca de un usuario de Zotero desde su ordenador personal. También incluye un manual de uso en el que se detallan los pasos necesarios para completar el objetivo del trabajo.

- Capítulo quinto: Conclusiones

  En este último capítulo se realizará un análisis del proceso de investigación y los resultados obtenidos en este trabajo. Asimismo, se lleva a cabo un estudio del proceso de investigación e implementación. Finalmente, existe una sección dedicada exponer los planes futuros del proyecto.

Capítulo 2

Contexto socio-económico y legal

Este capítulo se centra en estudiar la percepción en el entorno socio-económico de las tecnologías web y los gestores de referencias bibliográficas, así como el impacto del propio trabajo de fin de grado en este ámbito. También trata las leyes de protección de datos personales en general y las que regulan dicho trabajo. Finalmente, se explica el cálculo del presupuesto de este trabajo.

2.1. Aspecto socio-económico

En esta sección se comentarán los aspectos sociales y económicos de los conceptos tratados en este trabajo de fin de grado. Primero, se estudiarán las tecnologías web y redes sociales. Segundo, se analizará el propio trabajo en este entorno. Finalmente, se determinará la viabilidad del proyecto.

2.1.1. Impacto socio-económico de las tecnologías web y los gestores de referencias bibliográficos

En este apartado se hablará acerca del efecto de las tecnologías web y los gestores de referencias, incluidos en redes sociales para investigadores, en el entorno socio-económico actual.

Entorno social

Este subapartado está dedicado a comentar impacto social de las tecnologías web y los gestores de referencias bibliográficas.

- Tecnologías web

  Como se ha mencionado en la introducción del trabajo (ver apartado 3.2.1), las tecnologías web son todas aquellas que facilitan el hecho de compartir datos en la red. La tecnología web en la que se centra este trabajo son las APIs de tecnologías web. Las APIs ayudan a desarrolladores a conseguir autenticación y acceso a información de páginas web sin necesidad de interactuar con su código fuente.

  Estas tecnologías son usadas por todo tipo de páginas web con el objetivo de mejorar los servicios que ofrecen a sus usuarios y así conseguir una mejor experiencia de uso. Una función usual es la interacción entre redes sociales a fin de publicar la misma información en dos sitios web simultáneamente, agilizando el proceso. Más información sobre tecnologías web se encuentra en el apartado 3.2.

- Gestores de referencias bibliográficas

  Los gestores de referencias bibliográficas se han convertido en una herramienta fundamental en la comunidad científica. Actualmente existe una gran variedad de dichos gestores, algunos ejemplos son Zotero, ResearchGate y ORCID. Los gestores de referencias bibliográficas son analizados en profundidad en el apartado 3.1. Muchas de estas plataformas además de las herramientas de gestión de referencias, ofrecen funciones de red social con el objeto de hacerlas más modernas y atractivas. Dichas redes permiten a investigadores hacer que su trabajo tenga mayor visibilidad, consultar investigaciones de otros usuarios, referenciarlas, enviar mensajes, hacer preguntas y responderlas. Gracias a estos aspectos muchos gestores de referencias bibliográficas están experimentando un crecimiento exponencial.

Entorno económico

En este subapartado se hablará sobre el impacto económico de las tecnologías web y los gestores de referencias bibliográficas.

- Tecnologías web

  Tal y como se menciona previamente, las tecnologías web son utilizadas con la finalidad de compartir información en la red. Esta información no es solo enviada y almacenada, sino también analizada con diferentes fines. Una de las aplicaciones universales de las tecnologías web es el marketing. Se utilizan para obtener información sobre gustos y tendencias de los usuarios de la web con el propósito de: segmentar a usuarios en categorías según los datos recopilados, identificar tendencias en el mercado y anticiparse a cambios en este o realizar marketing especifico y así mostrar anuncios personalizados relevantes a cada usuario.

- Gestores de referencias bibliográficas

  Los gestores de referencias bibliográficas con funciones de red social actúan a modo de bolsa de empleo. Por un lado, ayudan a sus usuarios a obtener mayor visibilidad en su campo. Por otro lado, facilitan a empresas, laboratorios u organizaciones de investigación en el proceso de selección de empleados para sus proyectos según el perfil de usuario y ponerse en contacto con ellos. Por ejemplo, la red social Linkedin está fundamentada en este concepto.

2.1.2. Impacto socio-económico del proyecto

En esta sección se procederá a comentar el impacto socio-económico que puede generar el proyecto descrito en el apartado 1.1.1.

Entorno social

En un contexto social, la creación de una plataforma virtual desde la que un investigador sea capaz de administrar referencias en varios gestores de referencias bibliográficas al mismo tiempo, supone una mejora en la calidad del proceso de investigación. Primero, permite ahorrar tiempo y segundo, lo hace más accesible a personas sin experiencia en este campo.

Entorno económico

La automatización del proceso de gestión de referencias mediante una aplicación, ahorrará tiempo al investigador que la utilice. Este tiempo puede ser empleado en otros aspectos del proceso de investigación, mejorando así la eficiencia de su trabajo y reduciendo costes. En conclusión, la nueva plataforma beneficiará tanto al propio investigador como a la organización a la que pertenezca.

2.1.3. Viabilidad del proyecto

En este apartado se procederá a estudiar la viabilidad del proyecto final descrito en el apartado 1.1.1. Como punto de partida, se puede prever que la posibilidad de completar este proyecto es factible.

En la sección 1.2 son descritos algunos de los obstáculos que habrá que sobrepasar si se desea incluir los gestores de referencias ResearchGate, ORCID y Linkedin en el proyecto final. En el caso de ORCID, la solución se trata de conseguir acceso a todas las funciones de la API por medio de un miembro Premium, en la sección 3.1.2 se puede encontrar más información acerca de este tema.

A la hora de la integrar la plataforma Linkedin, deben obtenerse los permisos necesarios para la escritura de datos; bastará con pedir autorización a sus desarrolladores (ver aparatado 3.1.3 para más información)..

La implementación de ResearchGate presenta un mayor grado de dificultad. Debido a la ausencia de una API pública, solo cabría esperar a que los desarrolladores de ResearchGate publiquen las herramientas necesarias (ver aparatado 3.1.1 para más información)..

Como puede observarse los obstáculos que se encontrarán en el desarrollo del proyecto final pueden ser solucionados con tiempo. Asimismo, no se debe olvidar atenerse siempre a la legislación aplicable (apartado 2.2) y a las restricciones de uso de cada gestor de referencias.

2.2. Marco regulador

En este apartado se describirán las leyes, reglamentos y directivas que gobiernan la viabilidad del objetivo final de este trabajo. Puesto que este proyecto trata sobre la gestión de datos personales de usuarios en la web, se deberán estudiar lo siguiente. Primero, se comentará la regulación aplicable a la protección de datos personales en Internet, para después hablar acerca de los organismos que aplican dicha legislación.

2.2.1. Regulación aplicable

En este apartado se comentarán las normativas aplicables a la protección de datos en Europa y en España.

- **Artículo 18.4 de la Constitución Española (1978)**

  Este artículo de la constitución española indica que "la ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos"[1].

- **Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal**

  Tal y como se establece en artículo primero "La presente Ley Orgánica tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar"[2].

- **Reglamento Europeo (UE) 2016/679, de 27 de abril de 2016**

  El el artículo 95 se establece lo siguiente "El presente Reglamento no impondrá obligaciones adicionales a las personas físicas o jurídicas en materia de tratamiento en el marco de la prestación de servicios públicos de comunicaciones electrónicas en redes públicas de comunicación de la Unión en ámbitos en los que estén sujetas a obligaciones específicas con el mismo objetivo establecidas en la Directiva 2002/58/CE"[3].

- **Reglamento general de protección de datos 25 de mayo de 2018**

  Este reglamento trata el derecho al olvido digital. Se estipula que toda persona tiene derecho a eliminar sus datos personales cuando la finalidad para la que fueron recogidos haya finalizado. Asimismo, también existe derecho al olvido cuando el consentimiento no haya sido revocado o tenga lugar un tratamiento ilícito de datos con respecto a las normativa de la Unión Europea de protección de datos [4].

Estas son las directivas y reglamentos que componen el marco regulador de las comunicaciones electrónicas en la comunidad europea [5] [6].

- Directiva 2002/20/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, relativa a la autorización de redes y servicios de comunicaciones electrónicas (Directiva autorización).

- Directiva 2002/19/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, relativa al acceso a las redes de comunicaciones electrónicas y recursos asociados, y de su interconexión (Directiva acceso).

- Directiva 2002/22/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, relativa al servicio universal y los derechos de los usuarios en relación con las redes y los servicios de comunicaciones electrónicas (Directiva servicio universal).

- Directiva 2002/58/CE del Parlamento Europeo y del Consejo, de 12 de julio de 2002, relativa al tratamiento de los datos personales y a la protección de la intimidad en el sector de las comunicaciones electrónicas (Directiva sobre la privacidad y las comunicaciones electrónicas).

- Reglamento Nº 1211/2009 del Organismo de Reguladores Europeos de las comunicaciones electrónicas (ORECE), de 25 de noviembre de 2009.

- Reglamento Nº 531/2012 del Parlamento Europeo del Consejo, de 13 de junio de 2012, relativo a la itinerancia de las redes públicas de comunicaciones móviles en la Unión Europea.

2.2.2. Organismos reguladores

En este apartado se hablará brevemente de los organismos encargados de la protección de datos en España y Europa, respectivamente.

Cabe mencionar la existencia de la Agencia Española de Protección de Datos. Esta institución se encarga de controlar las infracciones que afectan a la protección de datos e impartir sanciones; tanto en el ámbito digital como físico (papel). España fue tardía en adaptarse a la legislación europea en cuanto a la protección de datos con carácter personal. Aun así, actualmente es uno de los países europeos con las normas más restrictivas en esta materia y un gran índice de imposición de sanciones [7] [8].

Por otro lado, a nivel europeo destaca el Grupo de Trabajo del Articulo 29 de la Directiva de Protección de Datos 95/46/CE. El Grupo de Trabajo 29 es un organismo dedicado a la protección de las personas en cuanto al tratamiento de los datos personales. Dicha organización es de carácter consultivo e independiente. Sus funciones son descritas en el articulo 30 de la Directiva 95/46/CE así como en el articulo 15 de la Directiva 2002/58/CE. [9].

2.3. Presupuesto

Este apartado está dedicado a la estimación del coste total de la realización del trabajo de fin de grado. Con dicha finalidad se ha distinguido entre los costes del material usado para su compleción y los costes del personal implicado.

2.3.1. Costes materiales

Se entiende como por materiales todos aquellos incluidos en los gastos de los elementos físicos del proyecto que no son personas. Los elementos utilizados en este trabajo son los siguientes.

- Ordenador personal

  Un ordenador desde el cual poder acceder a Internet y programar el código necesario. Dado que los procesos que se llevan a cabo en el trabajo no son especialmente intensivos en los componentes del ordenador, no es necesario renderización de vídeo o simulación de elementos finitos; el ordenador podrá ser de gama media-baja.

- Espacio de trabajo

  Una mesa y silla en la que poder utilizar el ordenador.

- Software de programación en Python 3.7

  El programa utilizado fue PyCharm versión *Community Edition*, que es gratuita.

- Editor de LaTeX

  La memoria del trabajo ha sido redactada en el software online Overleaf; herramienta de redacción del formato LaTeX. Este editor dispone de una versión sin coste y un servicio de subscripción *Premium* que amplia la capacidad de almacenar archivos del perfil de usuario.

- Gastos derivados del uso del ordenador, como luz y acceso a Internet.

En la siguiente tabla están representan los gastos materiales del trabajo. Para ello se ha obviado el coste de los elementos ordenador, mesa y silla. Dichos elementos no son incluidos dado que la mayoría de personas disponen de ellos, o pueden ser accedidos es bibliotecas de manera pública. Además se considera su depreciación como nula.

Tabla 2.1: Tabla de presupuesto de material

Descripción	**Coste Mensual (€)**	**Coste Diario (€)**	**Dedicación (meses)**	**% Uso diario**	**Coste imputable (€)**
Gastos de luz	135.73	4.38	4	2	10.51
Gastos de Internet	64.5	2.15	4	100	258
TOTAL					268.51

La ecuación 2.1 se utiliza en el calculo del coste imputable.

$$A = B * 30 * C * \frac{D}{100} \tag{2.1}$$

Donde: A representa el coste total de un elemento; B, el coste diario del uso del elemento; C, el número de meses durante los que se usó el elemento; y D, porcentaje del uso dedicado diariamente.

2.3.2. Costes de personal

Los costes de personal se calculan a partir del valor de las horas de trabajo de las personas implicadas en este trabajo. En el desarrollo de trabajo solo participaron dos personas. Por un lado el tutor encargado de supervisar, dirigir y revisar el trabajo de fin de grado, Juan Carlos González Víctores, y el autor del trabajo, Víctor Díaz Obregón, encargado de llevar a cabo el trabajo y redactar la memoria.

El tutor del trabajo es doctor en Robótica e Inteligencia Artificial así como Investigador en *RoboticsLab* de la Universidad Carlos III de Madrid. El autor es estudiante de cuarto curso del Grado en Ingeniería en Tecnologías Industriales en la Universidad Carlos III de Madrid. Con el objetivo de realizar el cálculo del coste del trabajo del tutor, se utilizará el precio de una hora de trabajo de un Ingeniero Senior (35 €/hora aproximadamente). Para el cálculo del coste del trabajo del autor, se considerarán sus horas de trabajo como las de Ingeniero Junior (15 €/hora aproximadamente).

En la siguiente tabla 2.2 se muestran los costes de personal del trabajo.

Tabla 2.2: Tabla de presupuesto de personal

Nombre y Apellidos	**Categoría**	**Precio (€/hora)**	**Horas de trabajo**	**Sueldo estimado (€)**
Juan Carlos González Víctores	Ingeniero Senior	35	50	1750
Víctor Díaz Obregón	Ingeniero Junior	15	352	5280
TOTAL				7030

2.3.3. Costes totales

En tabla 2.3 se representa el valor del presupuesto total estimado de este trabajo. Este ha sido calculado a parir de los gastos de personal y los gastos de los materiales utilizados.

Tabla 2.3: Presupuesto total aproximado del trabajo

Tipo de coste	**Presupuesto de costes totales (€)**
Costes materiales	7030
Costes de personal	268.51
TOTAL	7298.51

Capítulo 3

Estado del arte

En este capítulo se introducen los conceptos en los que se fundamenta la investigación realizada en este trabajo de fin de grado. Se estudian los gestores de referencias bibliográficas con funcionalidad de redes sociales para investigadores. Con dicha finalidad ello se analizarán cuatro plataformas: ResearchGate, ORCID, Linkedin y Zotero. Asimismo, este capítulo incluye un apartado dedicado al estudio de tecnologías web, donde se definirán los conceptos de API, mensajes HTTP, flujos de autorización OAuth y respuestas de un servidor durante la comunicación.

3.1. Estudio de gestores de referencias bibliográficas

Este apartado está dedicado al estudio de gestores de referencias bibliográficas con aspectos de redes sociales empleados por la comunidad científica.

Las redes sociales tienen un papel fundamental la sociedad actual. Ayudan a sus usuarios a relacionarse con otros miembros, descubrir personas con intereses, objetivos o gustos afines, compartir información de todo tipo, coordinar acciones, empezar movimientos sociales o crear conciencia. En general han hecho la vida de las personas más fácil pues elementos como libretas de teléfonos, agendas y tarjetas de visita han pasado todos a estar contenidos en un mismo lugar de fácil acceso [10].

Existe una gran variedad de redes sociales, desde Facebook o Twitter, hasta otras redes especializadas como Linkedin. Dicha red ha creado una nueva manera de interactuar en las relaciones profesionales, sustituyendo a las tradicionales tarjetas de visita, y poco a poco al viejo curriculum.

Figura 3.1: Redes sociales[1]

Teniendo en cuenta todas las ventajas que ofrecen las redes sociales y su gran variedad, es evidente que la comunidad científica adapte dichas funcionalidades a las plataformas de gestión de referencias bibliográficas.

Dichas plataformas están destinadas a personas que participan activamente en proyectos de investigación. Estos gestores permiten poner en contacto a científicos de todo el mundo y todo tipo de áreas de investigación. Son usados con el objetivo de compartir resultados de investigaciones, recursos como referencias bibliográficas, enlaces y documentos, recibir realimentación o *feedback* y aumentar la visibilidad de los propios investigadores y sus trabajos. Al igual que las redes sociales mencionadas anteriormente, el objetivo principal es poner en contacto a investigadores y compartir sus recursos, para crear una comunidad científica colaborativa y no competitiva.

Además de los servicios mencionados, muchos de estos gestores ofrecen herramientas de la búsqueda de trabajo. Para ello ponen en contacto a compañeros y ex-compañeros de trabajo. Asimismo disponen bolsas de empleo que muestran ofertas de trabajo relacionadas con la formación e investigaciones del usuario.

Los gestores de referencias bibliográficas con aspectos de redes sociales son una herramienta muy útil en la ciencia que cada vez está más presente en la sociedad. Este trabajo se centra en el estudio del papel de dichos gestores, para ello se analizarán las siguientes plataformas: ResearchGate, ORCID, Linkedin y Zotero.

[1] Imagen obtenida en: El Blog de Jose Facchin, "Las Redes Sociales más importantes del Mundo "Lista actualizada al 2018"". [En línea] Disponible en: `https://josefacchin.com/lista-redes-sociales-mas-importantes-del-planeta/` (último acceso 5 de agosto de 2018)

Las aplicaciones previamente mencionadas permiten a sus usuarios organizar, editar, almacenar y compartir referencias. Una referencia bibliográfica, es una serie de informaciones que posibilitan identificar una publicación, o un fragmento de la misma de manera sencilla; ya sea una página web, un vídeo, un libro o un capítulo de una revista.

Figura 3.2: Gestores de referencias bibliográficas[2]

Existen diversas modalidades a seguir formar una referencia bibliográfica dependiendo del tipo de referencia y del estilo de la misma, los datos que especifican la fuente de información, el orden y la estética varían. Cambian no solo la referencia en sí, sino también cómo aparecen a lo largo del documento cuando se las menciona. Existen muchos estilos de referencias como APA, empleado en educación, psicología y derecho o el estilo IEEE [11].

3.1.1. ResearchGate

En esta sección se hablará de los diferentes servicios que ofrece ResearchGate a sus usuarios y las herramientas para desarrolladores; donde se comentará el estado de su API y la conectividad con otras aplicaciones.

[2] Imagen obtenida en: Social media en investigación, "Ventajas e inconvenientes de las redes sociales temáticas para investigadores". [En línea] Disponible en: `https://socialmediaeninvestigacion.com/ventajas-redes-sociales-tematicas/` (último acceso 30 de julio de 2018)

Figura 3.3: Logo ResearchGate [3]

Descripción ResearchGate

ResearchGate es una herramienta de gestión de referencias bibliográficas más importantes del momento que cuenta con aspectos de red social. Fue fundada en 2008 por Ijad Madish y Sören Hofmayer y se define a si misma como una plataforma "creada por científicos, para científicos". Desde entonces ha crecido hasta tener más de 9 millones de usuarios en la actualidad y una colección de 80 millones de documentos [4].

Este gestor ofrece a sus usuarios una gran variedad de funciones. En primer lugar, permite subir documentos académicos como artículos, presentaciones o colecciones de datos y que sean almacenados. Estas publicaciones podrán compartirse con el resto de clientes y añadir colaboradores a las mismas que una vez verificados aparecerán en ellas. En segundo lugar, existe un servicio dedicado a indicadores bibliométricos llamado *RG Score*. Este contiene estadísticas personales únicas a cada investigador que recogen datos importantes de uso, como numero de visitas, descargas, citas a documentos o a sus autores. Estas estadísticas se muestran en forma de gráfica, que resulta muy útil a la hora de observar el progreso de un investigador a lo largo del tiempo e identificar tendencias y corregir errores.

Uno de los puntos principales de una red social es poner en contacto a sus usuarios, ResearchGate permite a sus usuarios interactuar con otros miembros de su plataforma para plantear dudas, obtener respuestas y encontrar soluciones a problemas de investigación. Además, un usuario puede conectar con otros investigadores creando así una red de contactos. Finalmente, ResearchGate posee un apartado de trabajo en el que se listan una colección de empleos personalizados a partir del perfil de cada usuario.

[3] Imagen obtenida en: ResearchGate, "Sales Executive - Scientific Advertising (m/f/d)". [En línea] Disponible en: `https://jobs.lever.co/researchgate/f37ad0d4-d42e-4ee9-b36d-317ed5032782?lever-origin=applied&lever-source=BerlinStartUpJobs` (último acceso 30 de julio de 2018)

[4] ResearchGate, "ResearchGate: About" [En línea] Disponible en `https://www.researchgate.net/about` (último acceso 15 de agosto de 2018)

Sin embargo, pese a las evidentes ventajas y servicios que ofrece ResearchGate, existen algunos inconvenientes relacionados con las políticas de comunicación y transparencia con sus usuarios. Asimismo, el diseño del indicador bibliométrico *RG Score* no refleja imparcialmente la calidad del trabajo de un investigador. Todo esto genera importantes dudas respecto a su uso con fines evaluativos y hacen que investigadores prefieran usar otras plataformas. [12].

API ResearchGate

En el momento de realizar este trabajo, ResearchGate no ofrece a sus usuarios una API pública a pesar de haber sido prometida por sus desarrolladores en 2011 [5]. La única posibilidad que existe actualmente de compartir información con otros servicios es conectar una cuenta con Facebook, Google o Linkedin; para ello es necesario dirigirse al aparatado *Connect with Services* en los ajustes de usuario y acceder a los diferentes enlaces de cada página.

3.1.2. Orcid

A continuación, se comentarán las funciones que ofrece ORCID a sus miembros, cómo se diferencia del resto de gestores de referencias bibliográficas y las funciones de su API.

Figura 3.4: Logo ORCID [6]

[5]Quora, "Does ResearchGate offer an API for external service or social communities in order to crosspost content from Facebook i.e. to ResearchGate?". 16 de septiembre de 2011[En línea] Disponible en: https://www.quora.com/Does-ResearchGate-offer-an-API-for-external-service-or-social-communities-in-order-to-crosspost-content-from-Facebook-i-e-to-ResearchGate)

[6] Imagen obtenida en: WikipediA, "File:ORCID logo.svg". [En línea] Disponible en: `https://en.m.wikipedia.org/wiki/File:ORCID_logo.svg` (último acceso 30 de julio de 2018)

Descripción ORCID

ORCID es una red social y gestor de referencias bibliográficas que se define como una organización sin ánimo de lucro. Su objetivo es "crear un mundo donde todos los que participan en la investigación, becas y la innovación se identifican de forma única y se conectan a sus contribuciones a través de diferentes disciplinas, fronteras físicas y el tiempo". La estrategia de ORCID en cuanto a diferenciarse del resto plataformas es proporcionar una clave única a cada miembro que les conecta con sus actividades de investigación; entre ellas se encuentran: mantenimiento de los perfiles de investigación, presentaciones manuscritas, solicitudes de subvención y solicitudes de patentes. ORCID proporciona a sus usuarios dos servicios básicos que se mencionan a continuación [7].

- Registro en su página web que proporciona al investigador un identificador único. También permite mantener un registro de documentos en línea y gestión de actividades como añadir publicaciones, trabajos, estudios, experiencia profesional y acceder a los perfiles de otros investigadores y sus publicaciones.

- Acceso a una API con diferentes grados de funcionalidad dependiendo del tipo de miembro.

El registro ORCID incluye información sobre sus miembros. En este registro se puede encontrar el nombre, correo electrónico, estudios y actividades de investigación de cualquier investigador que posea un ID de ORCID. Dicho registro puede ser accedido por toda persona que posea un identificador en la plataforma con el objetivo de buscar información sobre otros miembros. En el registro personal pueden ser modificadas las actividades de un usuario. En cuanto a organizaciones, estas pueden convertirse en miembros para añadir a sus empleados y estudiantes al registro de ORCID. Las organizaciones miembros de pueden obtener identificadores para sus componentes, actualizar los registros de sus miembros y recibir actualizaciones.

A los grupos que financian investigaciones, tanto asociaciones profesionales como académicas, ORCID les ofrece vincular a sus científicos con su trabajo y los programas de financiación que proporcionaron apoyo. Por otro lado, ORCID permite a las organizaciones de investigación como universidades, empresas o laboratorios nacionales, reducir el proceso de mantener sus registros actualizados y proporcionar validación con actualizaciones de fuentes fiables. Finalmente, ORCID hace posible a editores agilizar el proceso de admisión de manuscritos, mejorar la gestión de las bases de datos de autores y revisores y conseguir mayor precisión en las búsquedas de repositorios de artículos basados en nombres.

[7]ORCID, "DISTINGUISH YOURSELF IN THREE EASY STEPS". 16 de septiembre de 2011[En línea] Disponible en:https://orcid.org/

ORCID diferencia sus usuarios dependiendo de la contribución monetaria que recibe por parte de la organización a la que pertenecen; los agrupa en *Public*, *Basic* y *Premium*. *Public* es la categoría de miembro más baja y se obtiene al crear una cuenta en ORCID. *Basic* y *Premium* corresponden a los miembros que pertenecen a una organización que colabora con ORCID económicamente y se diferencian en la aportación monetaria.

API ORCID

Atendiendo al tipo de miembro, ORCID ofrece varios tipos de API: *Public* API, *Basic Member* API y *Premium Member* API. Las funciones de los diferentes tipos de API se muestran a continuación. Debido al acceso restringido a todas las funciones de la API, en este apartado se tratarán las interacciones con la *Public* API, conseguir un ID de ORCID autenticado y leer datos públicos del registro de ORCID.

- Autenticar un ID de ORCID

  Con el objeto de poder obtener un ID de ORCID autenticado con el que acceder a recursos protegidos, el primer paso es registrar una aplicación con ORCID. Al crear esta aplicación se debe definir su nombre, una URL de la propia aplicación y una URL de redirección. De esta manera se obtiene un identificador de cliente o *Client ID* y un secreto del cliente o *Client Secret* que serán utilizados en futuras interacciones entre la aplicación creada y ORCID [8].

  El proceso de obtención del código de acceso *access token*, se realiza mediante el flujo de autorización OAuth 2.0 usando un *authorzation code*, que se describe en la sección 3.2.2 Las siguientes URLs son necesarias durante el flujo de autorización para OAuth 2.0 que se explica más adelante en 3.2.2:

  - URL de autorización: https://orcid.org/oauth/authorize
  - URL donde obtener el *access token*: https://orcid.org/oauth/token

- Leer datos públicos del registro

  ORCID permite acceder a su registro a través de aplicaciones externas con el objeto de leer información sobre sus usuarios [9]. Para ello se realizará una petición HTTP *GET* a la siguiente URL "https://api.orcid.org/v2.1/[ORCID ID]/[Endpoint]".

[8]ORCID, "BASIC TUTORIAL: GET AN AUTHENTICATED ORCID ID". [En línea] Disponible en: `https://members.orcid.org/api/tutorial/get-orcid-id` (último acceso 16 de agosto de 2018)

[9]ORCID, "BASIC TUTORIAL: READ DATA ON AN ORCID RECORD". 16 de septiembre de 2011[En línea] Disponible en: `https://members.orcid.org/api/tutorial/read-orcid-records` (último acceso 16 de agosto de 2018)

Donde el ORCID ID que se incluye en la URL pertenece al usuario del que se quiere obtener información y el *Endpoint* de la URL se refiere al tipo de dato se desea acceder. En la tabla 3.1 se muestran algunos de los tipos de *Endpoint*.

Tabla 3.1: *Endpoint* y su descripción para ORCID

Endpoint	**Descripción**
/record	Vista resumida del registro completo del investigador
/keywords	Palabras clave relacionadas con la investigación
/address	País del investigador
/email	Email del investigador
/activities	Resumen de actividades del investigador

En el encabezado o *header* de la petición se deben incluir los siguientes parámetros:

- Accept: application/vnd.orcid+xml
- Authorization type: Bearer
- Access token: código de acceso conseguido en el apartado

	Public API	Basic Member API	Premium Member API
READING ORCID DATA			
Get authenticated ORCID iD	✓	✓	✓
Search/retrieve public data *ORCID iDs & data made public by iD holders*	✓	✓	✓
Search/retrieve member-subscriber data *Subject to permissions granted by iD holders*		✓	✓
WRITING ORCID DATA			
Add to and update ORCID records *Subject to permissions granted by iD holders*		✓	✓
Facilitated creation of new ORCID Records		✓	✓

Figura 3.5: Funciones de la API de ORCID dependiendo del *member*[10]

[10] Imagen obtenida en: ORCID, "INTEGRATE WITH THE ORCID API". [En línea] Disponible en: `https://orcid.org/organizations/integrators/API` (último acceso 16 de agosto de 2018)

3.1.3. Linkedin

Este apartado está dedicado a Linkedin. En él serán expuestos los servicios que dicha red ofrece a sus usuarios y las herramientas de comunicación web disponibles.

Figura 3.6: Logo Linkedin[11]

Descripcion Linkedin

Linkedin fue creada en el año 2002 por Reid Hoffman y se lanzó al mercado oficialmente el 5 de mayo de 2003. La empresa cuenta con un modelo de negocio diversificado e ingresos provenientes de suscripciones de usuarios, publicidad y soluciones al al hora de selección de personal. El mes de diciembre de 2016 Microsoft completó la compra de Linkedin. [12].

La plataforma ofrece a sus miembros una extensa red de contactos construida mediante conexiones directas. Una persona puede introducirse en dicha red a través de un contacto mutuo, favoreciendo así la interactividad. Los usuarios pueden subir su currículum vítae y modificar su perfil con la finalidad de que se vean reflejadas su experiencia de trabajo y sus habilidades profesionales. Linkedin beneficia a sus miembros, tanto a usuarios particulares como empresas, de distintas maneras. En cuanto a personas particulares, estas pueden encontrar ofertas de puestos de trabajo, guardar aquellas que les gustaría solicitar, revisar el perfil de empresas para mejorar su conocimiento acerca de ellas, ampliar su red de contactos o ver quién y cuántos usuarios han visitado su perfil. En cuanto a empresas, cumple el cometido de base de datos en la que consultar información sobre personas durante nuevas contrataciones [13].

[11] Imagen obtenida en: WikipediA, "Archivo:Linkedin Logo.svg". [En línea] Disponible en: `https://es.m.wikipedia.org/wiki/Archivo:Linkedin_Logo.svg` (último acceso 20 de agosto de 2018)

[12] Linkedin, "Acerca de Linkedin". 16 de septiembre de 2011[En línea] Disponible en: `https://about.Linkedin.com/es-es` (último acceso 16 de agosto de 2018)

[13] Ciudadano 2.0, "Qué es Linkedin, cómo funciona y qué te puede aportar esta red social profesional". 16 de septiembre de 2011 [En línea] Disponible en: `https://www.ciudadano2cero.com/`

API Linkedin

Linkedin permite obtener información acerca de sus usuarios a través de aplicaciones externas. El primer paso en el proceso de acceder a los datos protegidos es obtener autorización. Una vez conseguidos los permisos, el usuario podrá realizar peticiones a los servidores de Linkedin a fin de subir y obtener información.

- Obtener código de acceso en Linkedin

  Con el propósito de obtener un código de acceso, llamado *access token*, que permita acceder a recursos protegidos, el primer paso es registrar una aplicación con Linkedin. Al crear esta aplicación se debe definir su nombre, una URL de la propia aplicación y una URL de redirección. De este modo se obtendrá un *Client ID* y *Client Secret* que serán utilizados en futuras interacciones entre la aplicación y Linkedin. Se ha de tener en cuenta que para que la aplicación pueda ser utilizada, debe ser previamente autorizada. Para ello se solicitará permiso al equipo de desarrolladores de Linkedin [14].

  El proceso de obtención del código de acceso o *access token* se realiza mediante el flujo de autorización OAuth 2.0, con la variación *authorzation code*, que se describe en la sección 3.2.2. Las siguientes URLs son necesarias en dicho flujo de autorización:

  - URL de autorización:
    https://www.Linkedin.com/oauth/v2/authorization
  - URL para obtener el *access token*:
    https://www.Linkedin.com/oauth/v2/accessToken

- Leer y escribir datos en Linkedin

  Es posible acceder a información protegida a través de aplicaciones externas autorizadas con el objetivo de leer y escribir datos en los perfiles de sus usuarios. Para ello se realizarán peticiones HTTP *GET* (obtener datos) y *POST* (subir datos) a la siguientes URLs [15] [16].

  - *GET* URL
    "https://api.Linkedin.com/v1/people/[:Endpoints]?format=json"

`Linkedin-que-es-como-funciona/` (último acceso 16 de agosto de 2018)

[14]Linkedin Developers, "Authenticating with OAuth 2.0". 16 de septiembre de 2011. [En línea] Disponible en: `https://developer.Linkedin.com/docs/oauth2` (último acceso 16 de agosto de 2018)

[15]Linkedin Developers, "Share on Linkedin". 16 de septiembre de 2011. [En línea] Disponible en: `https://developer.Linkedin.com/docs/share-on-Linkedin` (último acceso 16 de agosto de 2018)

[16]Linkedin Developers, "Getting started with the REST API". 16 de septiembre de 2011. [En línea] Disponible en: `https://developer.Linkedin.com/docs/rest-api` (último acceso 16 de agosto de 2018)

- *POST* URL
  "https://api.Linkedin.com/v1/people/~/shares?format=json"

El *Endpoint* de la URL *GET* se refiere al tipo de dato al que se quiere acceder; si este no se rellena, la respuesta contendrá todos los campos que existan. Algunos ejemplos de *Endpoint* se muestran en la tabla 3.2.

Tabla 3.2: *Endpoint* y su descripción para Linkedin

Endpoint	**Descripcion**
/firstName	Primer nombre del usuario
/headline	Titulo de una publicación
/lastName	Apellido del usuario
/url	URL del perfil del usuario

En el encabezado o *header* de la petición *GET* deben incluirse los siguientes parámetros con la finalidad de que la respuesta del servidor sea satisfactoria.

- Content-Type: application/json
- x-li-format: json

El encabezado o *header* de la petición *POST* debe contener los siguientes parámetros con el propósito de que la respuesta del servidor sea satisfactoria.

- Host: api.Linkedin.com
- Connection: Keep-Alive
- Authorization: Bearer (código de acceso obtenido en el apartado anterior)

3.1.4. Zotero

En este apartado se hablará acerca de Zotero, sus funciones como gestor de referencias bibliográficas y las herramientas de comunicación web que ofrece a sus usuarios.

Figura 3.7: Logo Zotero[17]

Descripción Zotero

Zotero es una plataforma de gestión de citas bibliográficas lanzada en 2006. Se define a si misma como una herramienta de investigación de nueva generación. Se diferencia de otros gestores de referencias en que dispone de una aplicación dedicada en el escritorio y una extensión en el navegador web desde las que los usuarios pueden administrar su trabajo.

Gracias a dicha extensión, desde cualquier navegador web Zotero puede detectar el tipo de documento académico el usuario está viendo y automáticamente generar una cita bibliográfica de este, incluyendo título, autor o año de publicación. Las citas generadas son subidas a la biblioteca de investigación de la cuenta del usuario. Allí son almacenadas y sincronizadas con el objeto de poder ser usadas a través de varias plataformas.

Además, las referencias bibliográficas pueden ser generadas manualmente desde la URL "www.zotero.org" o en la aplicación de escritorio. Para ello se elegirá el tipo de documento académico que quiere generar de una lista proporcionada por Zotero y se rellenarán los campos que aparecen con los datos de la publicación que se desea citar.

Otra manera de añadir referencias bibliográficas a un registro es introduciendo el DOI (Digital Object Identifier o Identificador de Objeto Digital) de un documento académico. Un DOI es una cadena alfanumérica asignada a objetos digitales con el propósito de hacer más simple su identificación. Cada DOI es único y, una vez asignado a un elemento, se mantiene constante sin cambiar incluso cuando el objeto cambia de URL. Zotero añadirá a la biblioteca del usuario la publicación asociada al DOI introducido.

Una vez coleccionado el material, se puede acceder al mismo desde la página web en la URL "www.zotero.org" o la aplicación de escritorio. En esta pueden crearse nuevas bibliotecas y colecciones con el propósito de organizar las referencias como el usuario prefiera. Las numerosas funciones de búsqueda avanzada incluyen herramientas de minería de datos y la capacidad de guardar búsquedas (colecciones inteligentes) y etiquetas.

La biblioteca personal puede ser accedida remotamente desde cualquier lugar a través de Internet. Un usuario puede elegir generar copias de seguridad de su biblioteca en ubicaciones remotas. El propietario de una cuenta en Zotero puede crear y compartir colecciones con la finalidad de colaborar con otros usuarios de Zotero[13] [18].

[17] Imagen obtenida en: Zotero, "Zotero Brand". [En línea] Disponible en: `https://www.zotero.org/support/brand` (último acceso 17 de agosto de 2018)

[18] Zotero, "Your personal research assistant". 16 de septiembre de 2011. [En línea] Disponible en: `https://www.zotero.org/` (último acceso 17 de agosto de 2018)

API Zotero

Zotero ofrece a sus usuarios acceso a una API pública acompañada de una extensa documentación en la que se explican los servicios que esta ofrece. En este apartado se estudiarán las funciones de autenticación, leer datos y subir archivos.

- Autenticación

  Con el propósito de poder obtener un código de acceso llamado *API key* y autenticación que permita acceder a recursos protegidos, el primer paso es registrar una aplicación con Zotero. Al crear esta aplicación se obtendrá un identificador de cliente o *Client ID* y un secreto de cliente o *Client Secret* que serán utilizados en futuras interacciones entre la aplicación creada y Zotero.

  Los permisos de dicha aplicación pueden ser modificados al crearla o enviando los valores que se muestran a continuación en la cadena de consulta de la URL de autorización durante el protocolo OAuth.

  - *library_access*: Concede a la aplicación acceso a la biblioteca del usuario.
  - *notes_access*: Concede a la aplicación acceso a las notas del usuario.
  - *write_access*: Concede a la aplicación permisos de escritura.
  - *all_groups*: Concede a la aplicación acceso a todos los grupos.

  Las siguientes URLs son necesarias durante el flujo de autorización OAuth 1.0 que se explica más adelante en este documento (ver apartado 3.2.2) [19].

  - *Temporary Credential Request*: https://www.zotero.org/oauth/request
  - *Token Request URI*: https://www.zotero.org/oauth/access
  - *Resource Owner Authorization URI*: https://www.zotero.org/oauth/authorize

- Obtener datos de un usuario.

  Zotero permite realizar peticiones HHTP *GET* con el objeto de acceder a información. En estas peticiones debe especificarse el recurso al que se quiere acceder, el *User ID* de un usuario y enviarse el código de autorización obtenido anteriormente. La URL a la que se pedirá la información tiene la siguiente forma:

  "https://api.zotero.org/[users o groups]/[usersID o groupID]/[Endpoint]/"

[19] Zotero, "OAuth Key Exchange". 16 de septiembre de 2011. [En línea] Disponible en: `https://www.zotero.org/support/dev/web_api/v3/oauth` (último acceso 17 de agosto de 2018)

En la tabla 3.3 se muestran algunos de los recursos a los que se puede acceder al cambiar el parámetro *Endpoint* [20].

Tabla 3.3: *Endpoint* y su descripción para Zotero

Endpoint	**Descripcion**
/items	Vista resumida del registro completo del investigador
/items	Palabras clave relacionadas con la investigación
/items/trash	País del investigador
/email	Email del investigador

- Subir datos a la biblioteca de un usuario

  Zotero permite realizar peticiones *POST* a fin de subir documentos a la biblioteca de un usuario. En el cuerpo de estas peticiones debe de aparecer la referencia en formato JSON. Las plantillas de estas referencias pueden obtenerse con una petición *GET* a la siguiente URL:

  "https://api.zotero.org/items/new?itemType=[Tipo de documento]"

  El "tipo de documento" de la URL anterior debe rellenarse con un valor de una lista de documentos predefinida por Zotero como nota, libro o artículo. Realizando la petición *GET* a la URL completa se consiguen las plantillas de distintos tipos de datos. Esta plantilla será rellenada con el contenido de la referencia a subir. La URL a la que se hará la petición HTTP *POST* con la finalidad de subir el documento tiene la siguiente forma:

  "https://api.zotero.org/[users o groups]/[usersID o groupID]/items/"

  En esta URL se determinará el tipo de biblioteca a la que se desea acceder. En el caso de la biblioteca de un usuario se rellenará con *user* mientras que en el caso de la biblioteca de un grupo se completará con *groups*. Una vez elegido el tipo de biblioteca se incluirá el identificador correspondiente a esta. En el encabezado de la petición se añadirán los siguientes parámetros.[21].

  - Content-Type: application/json
  - Zotero-API-Key: API key

[20] Zotero, "Zotero Web API Item Type/Field Requests". 16 de septiembre de 2011. [En línea] Disponible en: `https://www.zotero.org/support/dev/web_api/v3/types_and_fields` (último acceso 17 de agosto de 2018)

[21] Zotero, "Zotero Web API Write Requests". 16 de septiembre de 2011. [En línea] Disponible en: `https://www.zotero.org/support/dev/web_api/v3/write_requests` (último acceso 17 de agosto de 2018)

3.2. Estudio de tecnologías web

En este apartado se hablará sobre las tecnologías web, más tarde se introducirá el concepto de API. A continuación, se procederá a comentar los distintos métodos de obtener la autorización que permita acceder a recursos protegidos en la red. Finalmente, se encuentra una sección dedicada a la descripción de las respuestas de un servidor durante la comunicación.

Las tecnologías web son todas aquellas que están relacionadas con el intercambio de información en la red. Las páginas web se caracterizan por contener todo tipo de información desde texto hasta sonido. En Internet existen herramientas, reglas y protocolos con el objetivo de facilitar el flujo de información. Por ejemplo, SVG o Gráficos Vectoriales Escalables es utilizado en el caso de describir imágenes como conjuntos de vectores y formas para ajustar su escala independientemente de su tamaño. Otros usos de tecnologías web son el Protocolo de Transferencia de Hipertexto o HTTP que define la transmisión de documentos hipermedia. Cabe mencionar el lenguaje de marcas de hipertexto o HTML, utilizado cuando se desea describir y definir el contenido de una página web en un formato bien estructurado. Este trabajo se centra en las interfaces de programación de aplicaciones o APIs. Estas son utilizadas con el objeto de facilitar la conexión entre servicios web y aplicaciones.

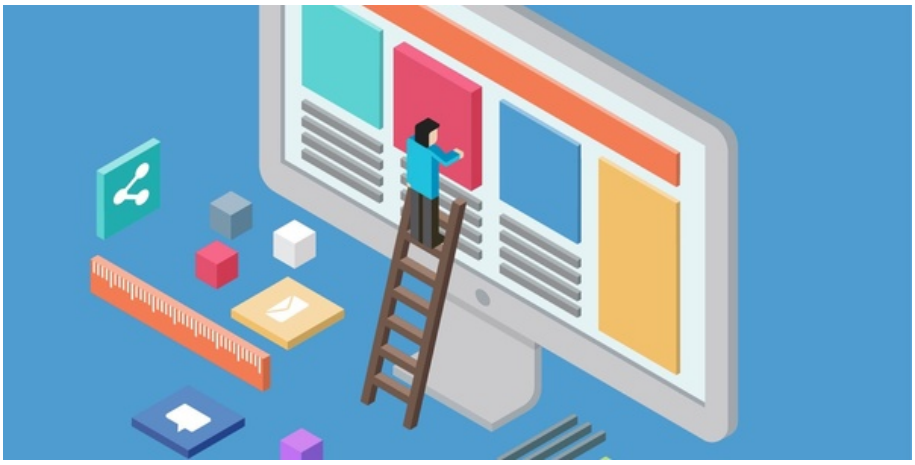

Figura 3.8: Tecnologías web[22]

[22] Imagen obtenida en: Youtube, "Las Tecnologías Web desarrollo Profesional Podcast 1 - Temporada 1". [En línea] Disponible en: `https://www.youtube.com/watch?v=y7gkSrgY4zI` (último acceso 18 de agosto de 2018)

3.2.1. API

En esta sección se hablará del concepto de API, los diferentes tipos de API, REST API, los mensajes HTTP y los distintos tipos de peticiones HTTP que se pueden realizar.

Definición

API es una sigla procedente del Inglés que significa Application Programming Interface o en español Interfaz de Programación de Aplicaciones. Desde el punto de vista de un desarrollador la conectividad con otras aplicaciones es un concepto muy importante, pues permite la integración de productos y servicios utilizando recursos ya creados. Una API es un conjunto de funciones que permite que aplicaciones separadas se conecten, comuniquen y compartan información con otras. La información que se comparte se conoce como recurso y la conectividad entre aplicaciones se llama integración.

Una API proporciona la posibilidad de acceder al contenido de la base de datos de una página web sin acceder a la misma directamente ni a su código fuente. El programador simplemente incorpora o programa las funciones necesarias de la API en el programa de la aplicación. Un programador puede usar un kit de desarrollo de software (SDK) u otra herramienta para desarrollar aplicaciones. Las herramientas de desarrollo pueden incluir APIs o permitir que el desarrollador tenga acceso remoto a las API.

Tipos de API

Existen tres tipos de API según estén fundamentadas en bibliotecas, en funciones de sistemas operativos o en servicios web [23] [24]. A continuación se describirán los tipos de API que existen.

- API de servicios web

  Las API de servicios web permiten intercambiar información entre una aplicación y un servicio web. Están basadas en estándares como SOAP, XML-RPC, JSON-RPC Y REST-HTTP y HTTPS. La información intercambiada es transmitida tanto en el cuerpo del mensaje, como en la cabecera del mismo, dependiendo del tipo de dato. En la cabecera se incluye información como nombres de usuario, palabras

[23] BBVA Open4U, "Qué es una API y qué puede hacer por mi negocio". 7 de junio de 2016. [En línea] Disponible en: `https://bbvaopen4u.com/es/actualidad/que-es-una-api-y-que-puede-hacer-por-mi-negocio` (último acceso 26 de agosto de 2018)

[24] FFEATHERS, "API types". Febrero de 2016. [En línea] Disponible en: `https:https://ffeathers.wordpress.com/2014/02/16/api-types/` (último acceso 26 de agosto de 2018)

clave que indiquen el tipo de mensaje que se intercambia e incluso códigos de acceso. Los formatos más extendidos para formar el mensaje son XML y JSON (ver figuras 3.9 y 3.10).

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema atributeFormDefault="unqualified"elementFormDefault="qualified">
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="points">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="point">
          <xs:complexType>
            <xs:attribute name="x"type="xs:unsignedShort" use="required"/>
            <xs:attribute name="y"type="xs:unsignedShort" use="required"/>
          <xs:complexType>
        <xs:element>
      <xs:sequence>
    <xs:complexType>
  <xs:element>
<xs:schema>
```

Figura 3.9: Ejemplo mensaje XML

```
payload = [
    {
        "itemType": "book",
        "title": title,
        "creators": [
            {
                "creatorType": "author",
                "firstName": first_name,
                "lastName": last_name,
            }
        ],
        "abstractNote": abstract,
        "place": school,
        "date": date,
        "language": language,
    }
]
```

Figura 3.10: Ejemplo mensaje JSON

- API de código fuente

  Las APIs de código fuente facilitan a una aplicación compartir bibliotecas y clases.

JavaScript es el medio en que se desarrollan las API de código fuente, como es frecuente en programación orientada a objetos. Uno de sus usos más importantes es en aplicaciones de cartografía como Google Maps. De este modo se permite a la aplicación de un servicio acceso a una biblioteca de mapas mundial y evitar crear un mapa del mundo desde cero.

- APIs de funciones en sistemas operativos

  Las APIs de funciones en sistemas operativos, son un método que utilizan los programas de software cuando deben interactuar con el sistema operativo. Están basados en un conjunto de funciones residentes en bibliotecas dinámicas. Un ejemplo es DirectX, que es una colección de APIs desarrolladas para facilitar las tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

Mensajes HTTP

Los mensajes HTTP consisten en peticiones o *requests* que un cliente realiza a cualquier servidor y respuestas o *responses* que estos servidores emiten de vuelta al cliente. Estos mensajes de petición y respuestas contienen información, que es pasada de un elemento a otro. Dichos datos compartidos pueden ser enviados de las siguientes formas [14].

- Encabezado del mensaje

  En el encabezado del mensaje o *header* se incluyen los parámetros necesarios para que el mensaje sea interpretado correctamente. Dichos parámetros pueden ser el tipo de aplicación, o las claves de autorización requeridas por un servidor. Cada campo de encabezado consiste de un nombre, que define el campo, seguido de dos puntos (":"), a la derecha de estos se colocará el valor del campo.

- Cuerpo del mensaje

  El cuerpo del mensaje o *body* corresponde a la parte del la petición o respuesta que contiene la información que se quiere compartir. Posibles ejemplos del contenido del *body* son una publicación nueva en una red social o una referencia bibliográfica. El cuerpo del mensaje está en un formato y codificación definidos en los campos del encabezado del mensaje. En la figura 3.10 se puede observar un ejemplo del cuerpo de un mensaje en formato JSON.

- Cadena de consulta de URL

En la Cadena de consulta o *query* de una URL pueden ser incluidos datos relevantes del mensaje. Los parámetros se colocarán al final de dicha URL después de signo de interrogación "?", si existieran más de un parámetro que enviar, estos irán separados por el símbolo "&". Un ejemplo de la este método de transmisión sería: "https://randomurl.org/?parameter_one=number&parameter_two=letter".

REST API

REST es el conjunto de principios de diseño y operaciones estándar como *GET* y *POST*, que se utiliza para coordinar interacciones entre aplicaciones, con el que los desarrolladores a través del protocolo HTTP pueden realizar solicitudes y obtener respuestas. El uso de HTTP permite la utilización de un amplio espectro de lenguajes de programación. Las peticiones se pueden realizar tanto en el formato XML como en formato JSON.

Los principios clave de REST implican separar su API en recursos lógicos. Estos recursos se manipulan utilizando solicitudes HTTP, con los siguientes métodos [15].

- *GET*: Petición de acceso a un recurso.
- *POST*: Petición de creación o actualización de un recurso.
- *PUT*: Petición idempotente para crear o actualizar un recurso.
- *PATCH*: Petición no idempotente ni segura para actualizar un recurso.
- *DELETE*: Petición de eliminación de un recurso.

Una REST API debe cumplir los siguientes requisitos [16]:

- La relación entre el cliente y el servidor es débil, a el servidor solo le importa la petición realizada y al cliente la información recibida del servidor. No es relevante para la otra parte cómo se han conseguido dichos datos o cómo se van a usar; son completamente independientes.

- Sin estado

  No hay necesidad de estados cada petición es independiente. Cada una de ellas contiene toda la información necesaria para ser procesada.

- Independiente

  La interfaz de la API debe permitir al cliente mandar una petición al servidor en un lenguaje independiente de los mismos.

- Cacheable

  Una REST API debe ser capaz de almacenar datos cacheables para la recuperación de datos frecuente o recientemente usados en el menor tiempo posible.

- Multicapa

  El servidor puede disponer de varias capas para su implementación. Que habilita disponer de un tope, a partir del cual el cliente no interacciona directamente con la arquitectura de software, esto se hace con el objetivo de mejorar la seguridad y el rendimiento.

3.2.2. OAuth

En este apartado esta dedicado a comentarán los flujos de autorización OAuth, como funcionan sus diferentes iteraciones, en que se diferencian y describirlos en profundidad.

Introducción

OAuth es un protocolo de autorización que permite establecer flujos de datos de manera segura con el objetivo de publicar e interactuar con datos protegidos, sin necesidad de exponer información sensible, como contraseñas, del usuario que quiere acceder a los datos. OAuth esta fundamentado en los protocolos HTTPS, utiliza códigos de acceso o *access tokens* para conceder autorización a APIs, dispositivos y aplicaciones en nombre de usuarios [17].

Hoy en día existen dos versiones de OAuth, OAuth 1.0 y su remplazo OAuth 2.0, totalmente diferentes y sin retrocompativilidad. Estas son las definiciones de la terminología usada en OAuth a las que más tarde se hará referencia.

- Usuario o *User*

  Entidad capaz de permitir acceso a recursos protegidos.

- Aplicación o *Client*

  Un sitio web o aplicación que utiliza OAuth para realizar peticiones HTTP autenticadas.

- Proveedor de servicio

  Un servicio web que permite el acceso a través de OAuth mediante un *Client*.

- Código de acceso o *access token*

  Un código usado por el *client* para obtener acceso a los recursos protegidos en nombre del usuario, en lugar de usar los credenciales del usuario del Proveedor de servicio.

- Recursos protegidos

  Datos controlados por el proveedor de servicio a los que el *Client* puede acceder una vez ha sido autenticado.

Obtener autenticación con OAuth 1.0

El método de autorización OAuth 1.0 es un proceso por el cual una aplicación o *client*, obtiene los permisos necesarios para acceder a recursos protegidos en nombre de un usuario o *User*. Este proceso se lleva a cabo sin necesidad de que el usuario comparta sus credenciales en el proveedor del servicio con la aplicación que quiere acceder a los datos. En lugar de credenciales como nombres de usuario y contraseñas, son utilizados códigos o *tokens* generados por el proveedor del servicio. Estos son los términos específicos del protocolo OAuth 1.0.

- *Request Token*

  Un código usado por el *Client* para obtener autorización del usuario e intercambiarlo por el *access token*.

- *Consumer Key*

  Un código usado por el *Client* para identificarse frente al proveedor de sericio.

- *Consumer Secret*

  Un código que utiliza el *Client* para acreditar propiedad del *Consumer key*.

- *Token Secret*

  Un código usado por el Consumidor para acreditar propiedad de un *token*.

- *OAuth Protocol Parameters*

  Parámetros que comienzan con el nombre oauth.

En OAuth 1.0 cada petición que realice la aplicación tiene que ser firmada y verificada por el proveedor de servicio, con el objetivo de prevenir que aplicaciones no autorizadas accedan a los datos protegidos. Existen varios métodos de firma HMAC-SHA1, RSA-SHA1, y PLAINTEXT, cada proveedor de servicio decide que método usar y debe especificarlo en la documentación de su API.

Además de la firma, las peticiones deben incluir identificadores llamados *timestamp* y *nonce*. El *timestamp* se define como el numero de segundos que han pasado desde la fecha 1 de Enero de 1970 a las 00:00:00, siempre tiene que ser positivo y mayor que el de la anterior petición. *Nonce* es una cadena o *string* aleatorio generado para cada petición, su función es asegurar que una petición no se ha realizado más veces y así mejorar la seguridad.

Es necesario aclarar que para poder llevar a cabo la autenticación con OAuth 1.0, el usuario debe generar una aplicación o *client* en el proveedor del servicio. Dicha aplicación accederá a los recursos protegidos en nombre del usuario. Al crear esta aplicación se obtendrán los parámetros llave de cliente o *client key* y secreto de cliente o *client secret*, que serán utilizados a lo largo del flujo con el objetivo de identificar a la aplicación. Además de la aplicación el usuario necesita las URL de petición, autorización y de emisión de códigos, que se encontrarán en la documentación de la API del proveedor de servicio.

A continuación se explicarán los distintos pasos del flujo de autorización de OAuth 1.0 (ver figura 3.11). Con la finalidad de mejorar la visualización se presentará la aplicación de cada proceso realizado a la plataforma Zotero. Como se observará a lo largo de la explicación el valor de los parámetros considerados sensibles (código de acceso) no serán representados por motivos de seguridad.

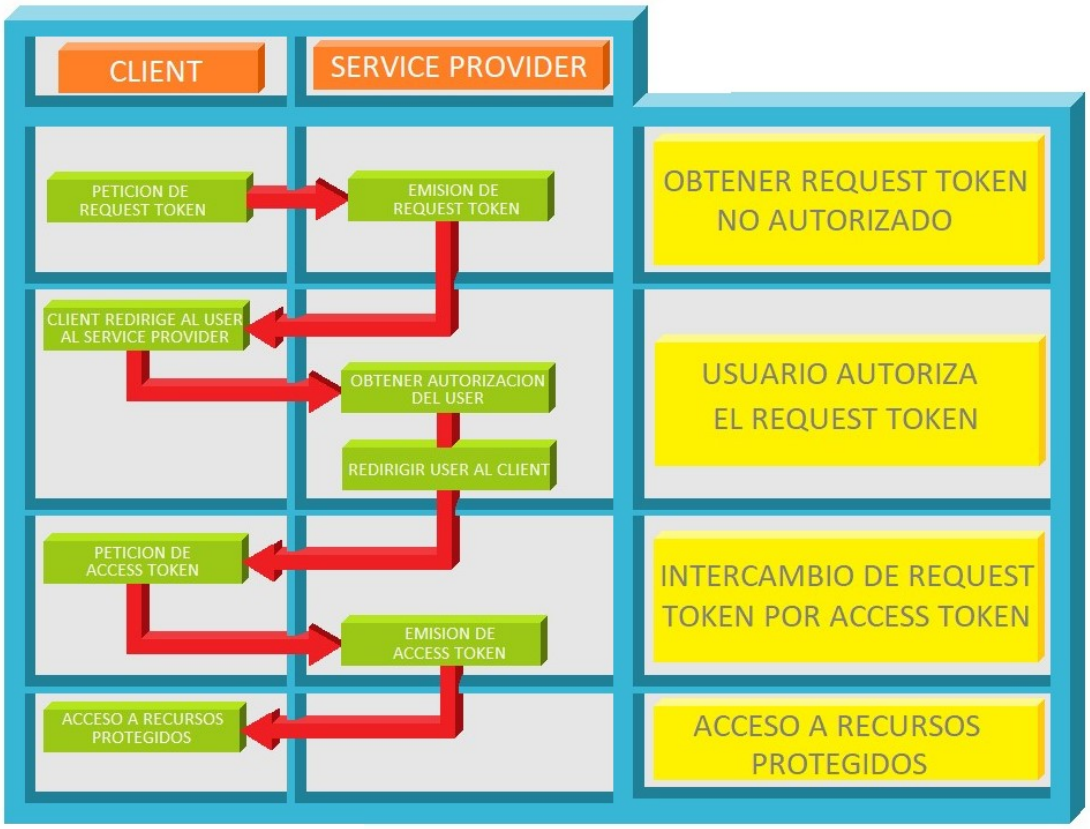

Figura 3.11: Flujo de autenticación para OAuth 1.0

1. La aplicación envía una petición de *request token*

   Al comienzo del proceso, la aplicación emite una petición HTTP *GET* al *Service Provider*. Esta *request* indica al proveedor del servicio que genere un *request token*, que posteriormente sera autorizado. La petición HTTP es enviada al URL de petición del proveedor del servicio. Un ejemplo de esta URL es:

   "https://www.zotero.org/oauth/request".

   Esta corresponde a la URL de petición del gestor de referencias Zotero. En la tabla 3.4, se puede observar la lista de parámetros que deben incluirse en el encabezado de la petición y un ejemplo de cada uno de ellos.

   Si la petición esta bien formada el Proveedor de servicio emitirá un *request token* y será enviado al *Client* en la respuesta HTTP del servidor. En la tabla 3.5 se encuentran los parámetros enviados en el cuerpo o *body* de la respuesta y ejemplos que complementan al los la tabla 3.4.

Tabla 3.4: Parámetros enviados en el encabezado de una petición de un *request token*

Parámetros	**Variable**	**Ejemplo**
ID del cliente	*oauth_consumer_key*	c9e4d318d612e32bc2b0
Tipo de firma	*oauth_signature_method*	HMAC-SHA1
Firma del cliente	*oauth_signature*	a6Evyaq5lZJ7GuTKjGo%3D
Identificador *timestamp*	*oauth_timestamp*	1534968499
Identificador *nonce*	*oauth_nonce*	iF4f8RZ85i5
Versión de OAuth	*oauth_version*	vacío
Parámetros adicionales	*Additional parameters*	vacío

Tabla 3.5: Respuesta a una petición de un *request token*

Parámetros	**Variable**	**Ejemplo**
Request token	*oauth_token*	ab5cef1fe533a88af421
Código secreto	*oauth_token_secret*	ab032ca8422739fc9939
Parámetros adicionales	*Additional parameters*	vacío

La variable *oauth_token* se corresponde con el *request token*. Si petición del *request token* es rechazada, el servidor deberá responder con el apropiado código de respuesta, correspondiente al error. Esta respuesta también puede incluir una breve descripción de porqué fue rechazada.

2. El usuario autoriza el *request token*

   La aplicación no podrá intercambiar el *request token* por el *access token* hasta que haya sido autenticado por el usuario. Con el objetivo de autenticarse el *Client* redirige al usuario a través del navegador al URL de autorización del proveedor del servicio. Un ejemplo de esta URL es:

   "https://www.zotero.org/oauth/authorize"

   Esta corresponde a la URL de autorización del gestor de referencias Zotero. En la cadena de consulta o *query string* de esta URL, el *Client* incluirá los siguientes parámetros.

   - *Request token*

     El parámetro llamado *oauth_token* obtenido en la respuesta de la petición del *request token*

   - URL de redirección

     El parámetro llamado *oauth_callback* que se definió al crear la aplicación en el proveedor del servicio.

La URL resultante de la inclusión de estos parámetros deberá parecerse a la siguiente URL de ejemplo:

"https://www.zotero.org/oauth/authorize?oauth_token=
ab5cef1fe533a88af421&oauth_callback=https://www.zotero.org/"

El usuario será redirigido a la URL mencionada anteriormente . Allí, el proveedor del servicio verifica la identidad del *Client* y pide consentimiento al usuario para autorizar al *Client* a acceder a los recursos protegidos.

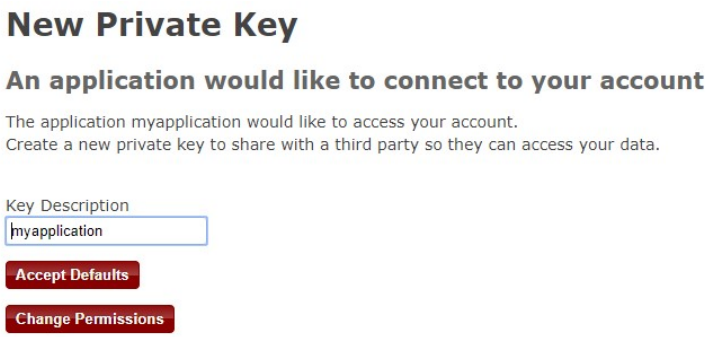

Figura 3.12: Ventana de autorización para Zotero

Si el usuario autoriza a la aplicación, el parámetro *request token* sera autorizado. El proveedor del servicio redirige al usuario a la URL de redirección que proporcionó el *Client* anteriormente.

En la cadena de consulta de la URL de la página en la que el usuario se encuentra tras la redirección, contiene los siguientes parámetros en su cadena de consulta.

- *oauth_token*
  Este parámetro es el *request token* anterior, pero ahora esta autorizado.

- *oauth_verifier*
  Un valor alfanumérico que se utilizará en el intercambio del *request token* por el *access token*.

3. El *Client* intercambia el *request token* por un *access token*

   Con la finalidad de acceder a los recursos protegidos se debe de intercambiar el *request token* autorizado por un *access token*. Para realizar el intercambio el *Client*

genera una petición HTTP *POST* que es enviada al URL de acceso del proveedor del servicio. Un ejemplo de esta URL es:

"https://www.zotero.org/oauth/access"

Esta corresponde a la URL de generación de códigos de acceso del gestor de referencias Zotero. En la tabla 3.7 son mostrados los elementos necesarios incluidos en encabezado.

Tabla 3.6: Petición de intercambio de *request token* por *access token* en OAuth 1.0

Parámetros	**Variable**	**Ejemplo**
Request token autorizado	*oauth_token*	ab5cef1fe533a88af421
ID del cliente	*oauth_consumer_key*	c9e4d318d612e32bc2b0
Tipo de firma	*oauth_signature_method*	HMAC-SHA1
Firma del cliente	*oauth_signature*	a6Evyaq5lZJmrhjcKjGo%3D
Identificador *timestamp*	*oauth_timestamp*	1534968500
Identificador *nonce*	*oauth_nonce*	ianemiRZ85i5
Versión de OAuth	*oauth_version*	vacío
Parámetros adicionales	*oauth_verifier*	3b611a41341d5cef2f1b
Parámetros adicionales	*Additional parameters*	vacío

Si la petición de intercambio esta bien formada el proveedor del servicio generará un *access token* y lo envía en el cuerpo o *body* de la respuesta al *Client*. Esta respuesta contiene los siguientes parámetros (ver tabla 3.7).

Tabla 3.7: Respuesta a la petición de un *access token* en OAuth 1.0

Parámetros	**Variable**	**Ejemplo**
Access token	*oauth_token*	Código de acceso
Código secreto	*oauth_token_secret*	Código secreto
Parámetros adicionales	*Additional parameters*	vacío

Tras completar el proceso de autenticación y haber conseguido el *access token*, el *Client* es capaz de acceder a recursos protegidos del proveedor de servicio. Las peticiones que realice la aplicación para acceder a dichos recursos deben incluir el *access token*.

Todas las peticiones realizadas al proveedor de servicio que sean rechazadas deberán ser respondidas con el apropiado código de error. Estas respuesta también puede incluir una breve descripción de porqué fue rechazada.

Obtener autenticación con OAuth 2.0

OAuth 2.0 es un estándar de autenticación que permite a una aplicación o *Client* obtener acceso limitado a recursos protegidos por medio de códigos llamados *access tokens*, en lugar de usar los credenciales de un usuario. Es necesario aclarar que para poder llevar a cabo la autenticación con OAuth 2.0, el usuario debe generar una aplicación o *client* en el proveedor del servicio. Dicha aplicación accederá a los recursos protegidos en nombre del usuario. Al crear esta aplicación primero, se deberán definir los permisos que el usuario dará a esta y proporcionará la URL de redirección necesaria durante la autenticación (para más información sobre creación de aplicaciones ver el ejemplo practico en el apartado 4.2). De este modo se obtendrán los parámetros llave de cliente o *client key* y secreto de cliente o *client secret*, que serán utilizados a lo largo del flujo con el objetivo de identificar a la aplicación. Además de la aplicación el usuario necesita las URL de autorización y de emisión de códigos, que se encontrarán en la documentación de la API del proveedor de servicio [18].

A continuación se explican los distintos pasos del flujo de autorización de OAuth 2.0 (ver figura 3.13). Con la finalidad de mejorar la visualización se presentará la aplicación de cada proceso realizado a la plataforma Linkedin. Como se observará a lo largo de la explicación el valor de los parámetros considerados sensibles (código de autorización y código de acceso) no serán representados por motivos de seguridad[19].

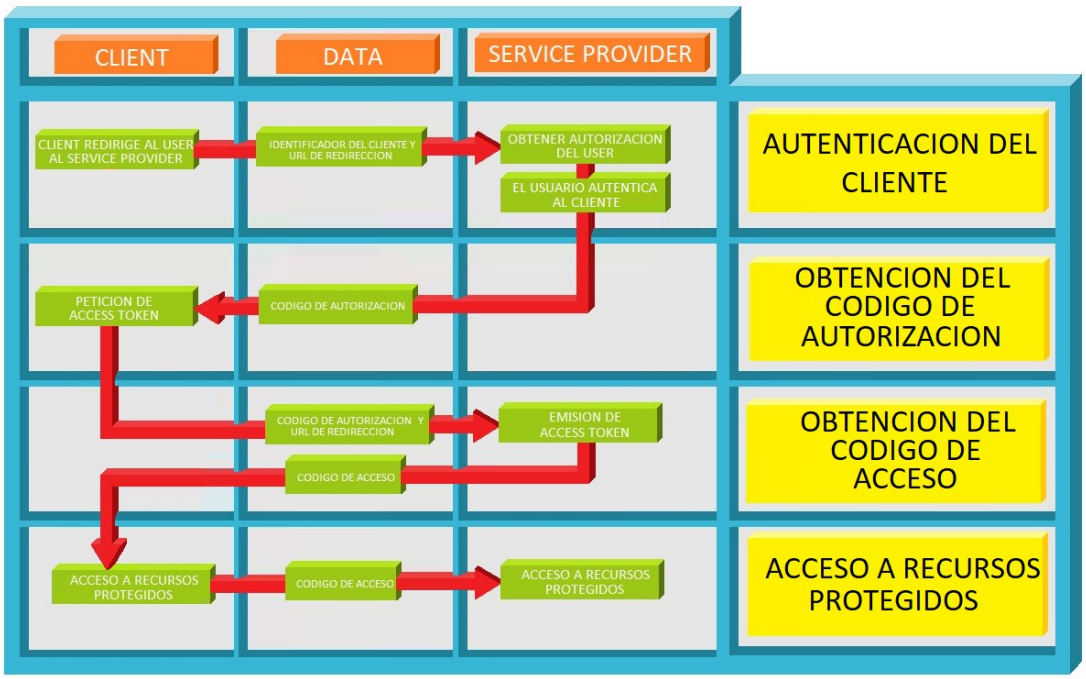

Figura 3.13: Flujo de autorización para el método *Authorization code*

1. Inicio del flujo

   La aplicación o *Client* necesita la autorización del usuario para obtener el *access token*. Existen varios procedimientos por los cuales un *Client* puede obtener esta autorización. Dichos métodos son: *Authorization code*, *Implicit*, *Resource Owner password credentials* y *Client credentials*. En este trabajo se estudiará el método *Authorization code*, por ser el de mayor uso en la autorización de aplicaciones en plataformas de gestión de referencias.

   El cliente inicia el flujo dirigiendo al usuario a través del navegador web a la URL de autorización. Un ejemplo de esta URL es:

   "https://www.Linkedin.com/oauth/v2/authorization"

   Esta corresponde a la URL de autorización de la red social Linkedin. En la cadena de consulta de esta URL se incluirán los parámetros mostrados en la tabla 3.8.

Tabla 3.8: Parámetros enviados en una petición de autorización en OAuth 2.0

Parámetros	**Variable**	**Ejemplo**
Tipo de autorización	*response_type*	code
ID del cliente	*client_id*	77k460xkcrc4un
Secreto del cliente	*client_secret*	sHsVFxBZNrm0ZNea
URL de redireccion	*redirect_uri*	https://www.Linkedin.com/feed/
Estado	*state*	987654321
Alcance	*scope*	r_basicprofile

2. El usuario autoriza a la aplicación

   El servidor de autorizaciones autentica al cliente (a través del navegador) y establece si el usuario concede o niega la solicitud de acceso del cliente. Para ello aparecerá en el navegador la siguiente ventana (ver figura 3.14).

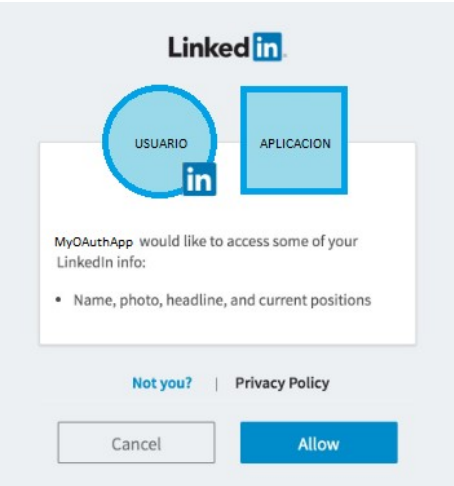

Figura 3.14: Ventana de autorización para Linkedin

3. Obtención de código de autorización

   Suponiendo que el usuario concede acceso, el servidor de autorización redirecciona al usuario de vuelta al cliente utilizando la URI de redirección proporcionada anteriormente. En la URI de redirección se incluye un código de autorización y estado local provisto por el cliente. Para extraer este código el cliente deberá parsear esta URL y aislarlo.

4. Formación de la petición del código de acceso

   Una vez que el cliente ha conseguido el código de autorización, este podrá solicitar un código de acceso o*access token* a la URL de emisión de *tokens* del servidor de autorización. Un ejemplo de esta URL es:

   "https://www.Linkedin.com/oauth/v2/accessToken"

   Esta corresponde a la URL de emisión de *access tokens* de la red social Linkedin. Para ello se llevará a cabo una petición HTTP *POST*. En el encabezado de la petición se incluirán los parámetros mostrados en la tabla 3.9.

Tabla 3.9: Parámetros enviados en una petición de código de acceso en OAuth 2.0

Parámetros	**Variable**	**Ejemplo**
Tipo de autorización	*grant_type*	*authorization_code*
Código de autorización	*code*	El codigo obtenido
ID del cliente	*client_id*	77k460xkcrc4un
Secreto del cliente	*client_secret*	sHsVFxBZNrm0ZNea
URL de redireccion	*redirect_uri*	https://www.Linkedin.com/feed/

5. Obtención del código de acceso

   El servidor de autorización autentica al cliente, valida el código de autorización, y asegura que el URI de redirección recibida coincide con el URI utilizado para redirigir al cliente en paso anterior. Si es válido, el servidor de autorización responde de nuevo con un token de acceso almacenado en la variable *access_token* y el tiempo por el cual será válido expresado en la variable *expires_in*. Opcionalmente, podrá incluir un token de actualización [20].

Con el *access token* obtenido por el método descrito anteriormente se podrán realizar peticiones con el objetivo de acceder a los recursos protegidos de un proveedor de servicio [19].

3.2.3. Códigos de respuestas de servidor

Los códigos de respuesta del servidor son enviados cada vez que se realiza una petición en la respuesta del servidor. Estos códigos ayudan a visualizar el estado de nuestra petición. El código de respuesta es un numero de tres dígitos, dependiendo del primer dígito se pueden clasificar en [14]:

- *Successful* 2XX

  Significa que la petición fue recibida, entendida y aceptada. A continuación se muestran algunos de los posibles códigos de esta categoría.

  - 200 OK: respuesta satisfactoria tras una solicitud exitosa.
  - 201 *Created*: aparece tras una petición aceptada de POST o PUT indicando que se ha subido el archivo a la página web de la API a la que se ha hecho la petición.
  - 204 *No Content*: respuesta satisfactoria que, típicamente, tiene lugar tras una operación de DELETE.

- *Redirection* 3XX

  Significa que el *Client* necesita realizar alguna acción complementaria para completar la petición.

  - 301 *Moved Permanently*: El recurso solicitado ha le ha sido asignado una nueva URL permanente y cualquier referencia futura a este recurso debe hacerse usando la nueva URL.
  - 302 *Moved Temporarily*: El recurso solicitado reside temporalmente bajo una URL diferente. Dado que la redirección puede ser alterada ocasionalmente, el cliente debería continuar usando el URI de solicitud en futuras solicitudes.
  - 302 *Not Modified*: Si el cliente ha realizado una solicitud *GET* y el acceso ha sido permitido , pero el documento no ha sido modificado desde la fecha y hora especificada en el campo *If-Modified-Since*, el *body* de la respuesta del servido debe estar vacío.

- *Client Error* 4XX

  Este código sera utilizado en los casos en que el cliente haya cometido un error . Si el cliente no ha completado el solicitud cuando se recibe un código 4XX, debe cesar inmediatamente de enviar datos al servidor.

  - 400 *Bad Request*: error de cliente debido a sintaxis no válida.
  - 401 *Unauthorized*: El cliente envió credenciales erróneos.
  - 403 *Forbidden*: El cliente no posee los permisos necesarios para realizar la petición.
  - 404 *Not Found*: El servidor no puede encontrar la respuesta demandada por el cliente.
  - 408 *Request Timeout*: Cuando el tiempo de espera del servidor acaba se produce este error.
  - 422: *Unprocessable Entity*: error de cliente en el cual la petición está bien formulada pero carece de semántica.

Capítulo 4

Desarrollo técnico

En este capítulo se describirá el proceso de implementación que permite subir referencias bibliográficas a la biblioteca de un usuario de Zotero desde su ordenador personal. También incluye un manual de uso en el que se detallan los pasos necesarios para subir una referencia bibliográfica a Zotero.

4.1. Implementación

En esta sección se hablará acerca del código utilizado en la realización de una petición POST a los servidores de Zotero con el objetivo de finalmente subir una referencia bibliográfica a la biblioteca de un usuario. Para ello se creará un código en el lenguaje de programación Python, que en conjunto con las acciones manuales del usuario, deberá cumplir las siguientes funciones.

- Obtener el ID de Zotero de la biblioteca del usuario a la que se subirá el archivo.
- Obtener la llave o *API key* para acceder a la biblioteca del usuario a la que se subirá el documento.
- Identificar el archivo que contiene la referencia bibliográfica y extraer información de este.
- Construir una petición POST que contenga todos los datos necesarios para actualizar la biblioteca del usuario.

4.1.1. Requisitos previos

El código fue creado en el lenguaje de programación Python utilizando la versión 3.7. La plataforma de desarrollo de APIs, Postman, fue utilizada con el objetivo de obtener una plantilla del código Python. A partir de dicha plantilla se elaborará el código final. El programa usado para la escritura, compilación y revisión de este código es JetBrains PyCharm Community Edition 2018.2. Todos los apartados descritos en esta sección se llevaron a cabo en el sistema operativo Windows.

La tecnología en la que se basan las funciones que debe cumplir el código es la API de Zotero. Dentro de este trabajo de fin de grado, en la sección 3.1.4, así como en la documentación disponible en [1] puede encontrarse información sobre esta API. Para llevar a cabo esta sección es necesario que el usuario posea una cuenta en el gestor de referencias Zotero y haya creado un aplicación en el mismo. Más detalles sobre este tema se explican a continuación y en la sección 4.2

4.1.2. Desarrollo del gestor de referencias

De acuerdo con las funciones del programa descrito anteriormente se puede dividir el proceso de implementación en los siguientes aparados.

Obtención del ID y *API key* de la biblioteca de usuario de Zotero

Aunque existe la posibilidad de automatizar el proceso de obtención de estos parámetros, en este proyecto se utiliza un método manual. La descripción detallada del proceso manual puede encontrarse en el apartado 4.2.

Obtención del código Python base en Postman

Para poder realizar este paso es necesario instalar el software Postman en un ordenador. Postman puede ser descargado de forma gratuita desde su página web, en la URL "https://www.getpostman.com/". Una vez instalado el programa el usuario se dirigirá al apartado de *File* y pulsará en *New*. Se abrirá una ventana con los diferentes tipos de proyectos que se pueden crear. El usuario elegirá *Request*. Tras rellenar los campos que

[1] Zotero, "Zotero Web API v3". 27 de noviembre de 2017. [En línea] Disponible en: https://www.zotero.org/support/dev/web_api/v3/start (último acceso 20 de agosto de 2018)

han aparecido se abrirá la ventana del proyecto creado. En esta ventana se encuentran las pestañas de: *Authorization*,*Headers*, *Body*, *Pre-request Script* y *Test*. A continuación se explicarán los cambios que se harán en cada una de ellas.

- Pestaña *Authorization*

  Esta es la pestaña en la que el usuario se encuentra al crear un nuevo proyecto, la pantalla tendrá un aspecto similar a la figura 4.1, pero no exactamente igual, para ello hay que completar las siguientes instrucciones.

  En el campo *Enter request URL* (rojo) se debe introducir la URL a la que se realizará la petición HTTP, en este caso es:

  "https://api.zotero.org/users/5050362/items"

  Esta es la URL a la se realizan peticiones en Zotero, para saber como se obtiene esta URL ir a la sección 3.1.4, en el apartado dedicado a APIs.

  Por defecto el tipo de petición (amarillo) es *GET* se debe cambiar este el campo a *POST*, ya que el objetivo es subir una archivo.

  En el campo *TYPE* (verde) se seleccionará el protocolo de autorización que usa el proveedor del servicio, en este caso es OAuth 1.0. Para identificar el protocolo de autenticación ir a la sección 3.1.4, en el apartado de si API.

  En el campo *Consumer Key* y *Consumer Secret* (azul) se rellenarán con los valores de este mismo nombre obtenidos al crear una aplicación en Zotero. El proceso de creación de una aplicación se encuentra en el apartado 4.2.

  En el apartado *ADVANCED* el campo *Signature Method* (morado) debe ser rellenado con el valor del método de firma especificado por el *Service Provide*, en este caso HMAC-SHA1. Para saber como se obtiene esta URL ir a la sección 3.1.4, en el apartado de si API y más información en la sección 3.2.2. Tras estos pasos la ventana de *Authorization* debería ser igual que la figura 4.1

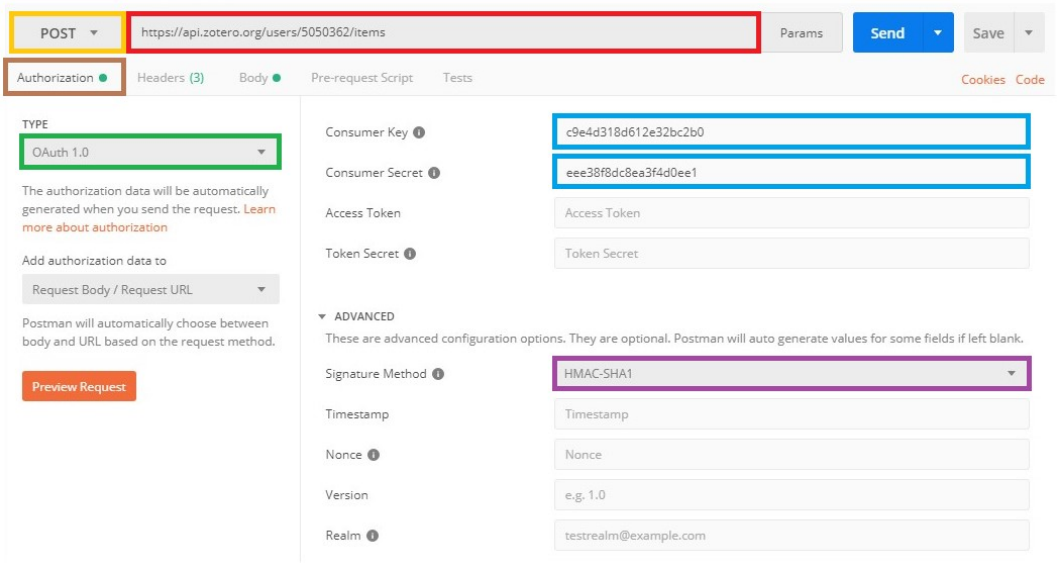

Figura 4.1: Ventana *Authorization* en Postman

- Pestaña *Headers*

  Cuando el usuario ha completado los campos de la ventana *Authorization*, pasará a la ventana de *Headers* (ver figura 4.2). En esta serán incluidos los valores que deberán ser enviados en el encabezado de la petición. En primer lugar el tipo de contenido que se enviará se define con "Content-Type: application/json" (amarillo) y en segundo lugar el código de acceso o *API key* en la forma "Zotero-API-Key: API key" (azul). Una vez incluidos estos datos la ventana *Headers* deberá ser igual que la figura 4.2.

Figura 4.2: Ventana *Headers* en Postman

- Pestaña *Body*

  En la pestaña de *Body* se incluirá el contenido que sera subido a la biblioteca de Zotero. Se seleccionará las opciones *raw* (amarillo) y *JSON (aplication/json)* (verde). En el contenido se pegará el cuerpo del mensaje en formato JSON que se quiera enviar (azul), en este caso es la plantilla de una "nota."[en] Zotero. Los motivos de enviar este contenido son, primero, se trata la plantilla más sencilla y segundo, una vez generado el código con Postman, habilitará a identificar la variable llamada *payload*. En principio, el contenido de esta variable será la plantilla de la nota y más tarde, será sustituido por la plantilla de "libro". Más adelante se explicará como obtener las plantillas de los elementos mencionados. Una vez incluidos estos datos la ventana *Body* deberá ser igual que la figura 4.3.

Figura 4.3: Ventana *Body* en Postman

Cuando se han completado todos los pasos mencionados anteriormente (ver figura 4.4), se procederá a enviar la petición pulsando el botón *Send* (verde). Cuando la petición haya sido enviada si todos los datos introducidos eran correctos, obtendremos una respuesta satisfactoria en formato JSON que incluirá los datos subidos. Ahora que se ha comprobado que la petición funciona, se procederá a generar el código Python que sera usado como base del programa. Para ello se pulsará el botón *Code* (rojo), aparecerá una ventana (azul) donde se debe seleccionar la opción *Python Request* (amarillo). El resultado debe ser igual a la figura 4.4

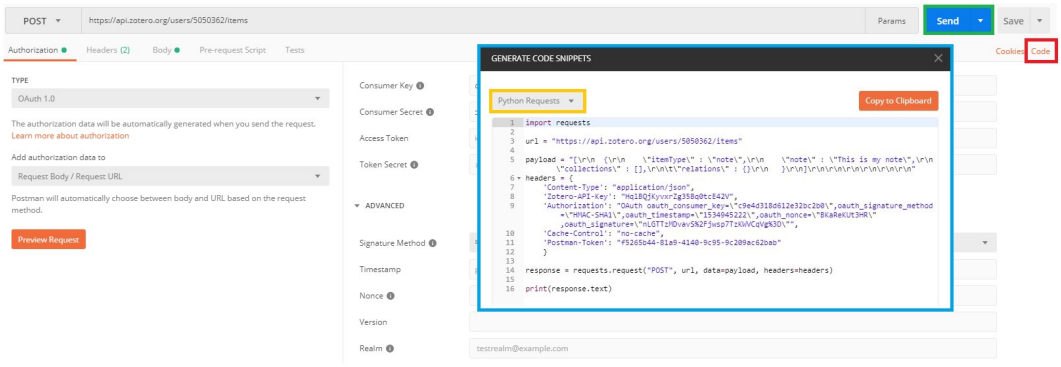

Figura 4.4: Obtención de un código Python en Postman

Ahora que se ha obtenido el código base, puede continuar su modificación en el programa JetBrains PyCharm Community Edition 2018.2 para ajustarlo al objetivo del trabajo.

Desarrollo del código de Python

A continuación se describirán los distintos elementos del código Python usado con la finalidad de construir y realizar una petición *POST* con el objetivo de subir una referencia a la biblioteca de un miembro de Zotero. Para ello se deben cumplir los requisitos descritos en la sección 4.1.1.

1. Importar los módulos necesarios y guardar el ID y *API key*.

   En primer se importarán los módulos que contienen las funciones y bibliotecas necesarias. Estos módulos deberán ser instalados previamente: *request*, *json* y *os*. A continuación se explica la función que cumple cada uno de ellos.

   - Módulo *request*

     Importar este módulo habilita a realizar peticiones HTTP/1.1 en Python. Estas peticiones pueden incluir contenido como *headers*, parámetros y datos, a través de librerías en Python. También permite añadir *query strings* a URLs y codificar el contenido de peticiones *POST* con el propósito de protegerlas [21].

   - Módulo *json*

JavaScript Object Notation o JSON es un formato de texto utilizado en la serialización de datos estructurados. JSON puede representar cuatro tipos de variables primitivas (cadenas de caracteres, números, booleanos y null) y dos tipos estructurados (objetos y vectores) [22].

- Módulo *os*

  El módulo *os* permite el acceso a funciones dependientes del sistema operativo, como conocer el directorio actual, cambiar de directorio de trabajo, crear y eliminar un directorio o eliminar un archivo [2].

Una vez completada la importación de los módulos necesarios, se procede a pedir al usuario los siguientes datos el identificador del usuario o *UserID* y el código de acceso o *API key* (para saber como el usuario obtiene estos parámetros ir a la sección 4.2). Los datos introducidos por el usuario son almacenados en las variables *user_id* y *api_key* en forma de cadena de caracteres o *string* para su posterior uso (ver figura 4.5).

```
1 user_id = str(input("Introduzca su numero de usuario: "))
2 api_key = str(input("Introduzca la API key ... "))
```

Figura 4.5: Código para conseguir las variables *user_id* y *api_key*

2. Localizar el archivo y parsearlo.

   El siguiente paso es localizar el archivo que será subido a la biblioteca del usuario. Para ello deben especificarse varias variables. Primero la localización del archivo en el ordenador del usuario Y segundo, el nombre del propio archivo. Dichos parámetros son guardados en la variables *path* y *file_name* respectivamente. La variable *ext* tiene la función de identificar el tipo de archivo al que se quiere recurrir, por ejemplo .txt o .bib (ver figura 4.6).

```
1 path = 'C:\\Users\\Victor\\Desktop\\TFG'
2 os.chdir(path)
3 file_name = 'morante2016thesis.bib.txt'
4 ext = file_name[-3::]
```

Figura 4.6: Código para localizar un archivo en Python

[2]LIBROSWEB, "10.1. Módulos de sistema". 16 de septiembre de 2011[En línea] Disponible en: http://librosweb.es/libro/python/capitulo_10/modulos_de_sistema.html

A continuación, ha de obtenerse información del archivo de texto localizado, esta información incluye datos como nombre del autor, lugar y fecha de publicación o el titulo de la publicación . Para ello se parseará el documento utilizando los comandos *split* y *partition*. La información obtenida es guardada en variables de tipo *string* para su posterior uso. En la figura 4.7 se pueden observar el fragmento de código utilizado con el objetivo de extraer el nombre del autor del archivo.

```
if ext == 'txt':
    with open(file_name) as f:
        content = f.readlines()
        for line in content:

            if 'author' in line:
                name_aux = line.split("author = {")[1]
                first_name_aux = line.split(", ")[1]
                first_name = str(first_name_aux.partition("}")[0])
                last_name = str(name_aux.partition(",")[0])
                print(first_name)
                print(last_name)
                print()
```

Figura 4.7: Fragmento del código para parsear un archivo en

3. Formación de la petición *POST*.

   La figura 4.8 muestra la URL a la que se enviará la petición *POST*. Esta URL se obtiene en la documentación de la API de Zotero disponible en 1. Dicha URL debe reflejar primero, el tipo de biblioteca a la que se subirá el archivo. Las opciones son *users*, en el caso la biblioteca de un usuario, y *groups*, en el caso la biblioteca de un grupo. Segundo, el número de la biblioteca del usuario o del grupo.

```
url = str("https://api.zotero.org/users/%s/items" % user_id)
```

Figura 4.8: URL a la que se realizará la petición *POST*

En la cadena de consulta o *query string* de la URL a la que se realizará la petición *POST*, se incluyen los parámetros que el servidor de autenticación de Zotero necesita a la hora de verificar que la petición que sea autorizada (ver figura 4.9). Si alguno de los datos no es correcto, el servidor devolverá de petición no autorizada error 403.

```
querystring = {"key": api_key,
               "oauth_consumer_key": "c9e4d318d612e32bc2b0",
               "oauth_signature_method": "HMAC-SHA1",
               "oauth_timestamp": "1533545627",
               "oauth_nonce": "2QoS1yK1sqA",
               "oauth_signature": "07H3vv7kNXPj7ni9ju0n59uisnk="
               }
```

Figura 4.9: Cadena de consulta de la petición *POST*

La carga o *payload*, será el cuerpo de la petición *POST*. En este caso contendrá la plantilla de referencia bibliográfica para un libro de Zotero (ver figura 4.11). Esta plantilla tiene la forma de un texto JSON y se obtiene realizando una petición *GET* a la siguiente URL "https://api.zotero.org/items/new?itemType=book.[en] el programa Postman(ver figura 4.10). La respuesta de esta petición es la plantilla de referencia de un libro. Del mismo modo que se ha obtenido la plantilla del elemento libro, se pueden conseguir otros cambiando el campo "item.[en] la sección "itemType=[item]"de la URL de la petición.

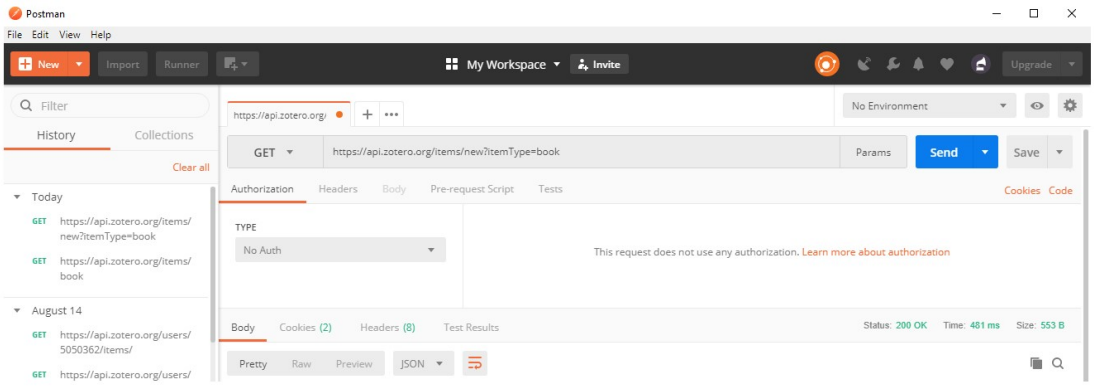

Figura 4.10: Obtener plantilla para un libro en *Postman*

La plantilla obtenida se incluye en la variable *payload*, que será el cuerpo de la petición y es rellenada con los datos obtenidos del parseado del documento, En la figura 4.11 se muestra una sección del cuerpo total. Esta carga es enviada bajo el apartado *data* de la petición *POST*. Si alguno de los datos no es correcto o la plantilla esta mal formada, el servidor devolverá error de JSON mal formado 403.

```
payload = [
    {
        "itemType": "book",
        "title": title,
        "creators": [
            {
                "creatorType": "author",
                "firstName": first_name,
                "lastName": last_name,
            }
        ],
        "abstractNote": abstract,
        "place": school,
        "date": date,
        "language": language,
    }
]
```

Figura 4.11: Código del fragmento del cuerpo de la petición *POST*

La cabecera o *header* de la petición HTTP permiten al cliente y al servidor enviar información que define a la petición o respuesta. En este caso se usan con el objetivo de especificar el tipo de contenido que es enviado y el uso del cache (ver figura 4.12).

```
headers = {
    'Content-Type': "application/json",
    'Cache-Control': "no-cache",
}
```

Figura 4.12: Código del encabezado de la petición *POST*

4. Enviar la petición *POST*.

Una vez definidos todos los campos anteriores, se procederá a enviar la petición *POST* HTTP a los servidores de Zotero (ver figura 4.13).

```
response = requests.request("POST",
                            url,
                            data=json.dumps(payload),
                            headers=headers,
                            params=querystring
                            )
```

Figura 4.13: Petición *POST*

5. Analizar la respuesta del servidor.

   Dependiendo de la respuesta del servidor(ver seccion 3.2.3), se han distinguido dos casos:

   - La petición estaba bien formada

     Tanto la sintaxis como los códigos de autorización necesarios eran válidos. En este caso el servidor devolverá el código *status code* igual a 200 y se imprimirá la respuesta que será la referencia bibliográfica en formato JSON (ver figura 4.14).

```
"successful": {
    "0": {
        "key": "9T3PPKDEP",
        "version": 167,
        "library": {
            "type": "user",
            "ide": 5050362,
            "name": "Victordiazobregon",
            "links": {
                "alternate": {
                    "href": "https://www.zotero.org/victordiazobregon",
                    "type": "text/html"
```

Figura 4.14: Fragmento de respuesta a una petición exitosa

   - La petición estaba mal formada

     En este caso el servidor devolverá el código *status code* que identifica al erro que se ha producido y se imprimirá la respuesta que será una breve descripción del error encontrado (ver figura 4.15).

```
403
Invalid key
```

Figura 4.15: Respuesta a una petición con errónea

4.2. Guía de uso

En esta sección se describirán todos los pasos necesarios que un usuario ha de seguir para replicar el objetivo de este TFG: conseguir autenticación y subir automáticamente una referencia bibliográfica a Zotero.

4.2.1. Creación de una cuenta en Zotero

El primer paso de este manual es la creación de una cuenta en la plataforma Zotero. Con el fin de subir referencias bibliográficas, el usuario deberá poseer un perfil en Zotero que le garantiza acceso a una biblioteca personal donde serán almacenadas. El usuario siguiendo este tutorial deberá dirigirse a la URL "https://www.zotero.org/user/register" y allí registrarse como un nuevo miembro. La figura 4.16 muestra los campos que es necesario completar durante el proceso de registro. Si la persona ya dispone de un usuario este paso no es necesario.

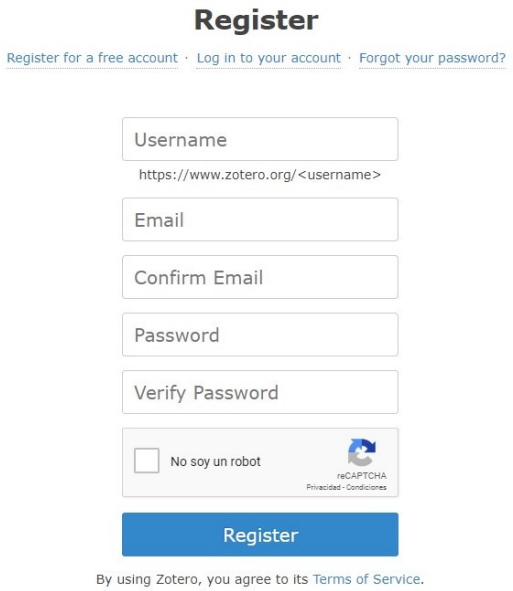

Figura 4.16: Pantalla de registro en Zotero

Una vez acabado el proceso de registro, el usuario ya cuenta con un perfil en Zotero. Este perfil le permite el acceso a los servicios de este gestor de referencias, incluida la disponibilidad de las herramientas de su API.

4.2.2. Creación de una aplicación en Zotero

El segundo apartado del manual trata sobre la creación de una aplicación en Zotero. Esta aplicación permite la interacción con la API de Zotero. Las funciones que se utilizarán, dentro de la lista de opciones que esta, ofrece son:

- Autenticación: Necesaria en el caso de acceder a recursos protegidos y gestionar información de la biblioteca de usuarios.

- Escritura: Necesaria con el objetivo de gestionar información de la biblioteca de usuarios y subir una referencia bibliográfica.

El primer paso en el registro de una aplicación será iniciar sesión en Zotero. Seguidamente, el usurario incluirá en la barra de búsqueda de su navegador la URL de la página principal de Zotero "https://www.zotero.org/" y escribirá "/oauth/app" al final de esta. Más adelante se justifica este paso, que a primera vista parece innecesario. Si la URL está bien formada, el usuario se encontrará ahora en la página que se muestra en la figura 4.17. Nótese que en la parte superior a la izquierda de la figura se indica el camino que el usuario seguiría en el acceso a esta sección. Según se indica en la figura un usuario podría desde el apartado *Home* (la página principal) dirigirse a *Settings* (los ajustes de su cuenta) y finalmente *OAuth Applications*. Realizar lo descrito anteriormente es imposible, pues no existe ninguna opción en *Settings* de la cuenta de un usuario que permita acceder a la URL de la figura 4.17; es por eso que se tiene que incluir directamente en el buscador. Esta URL también puede encontrarse en la documentación de la API.

Figura 4.17: Pantalla de administración de aplicaciones en Zotero

Una vez que el usuario se encuentra en esta ventana, se debe pulsar el botón *Register new application*, que llevará al usuario a la siguiente página (ver figura 4.18). Esta es la página desde la cual se generan las aplicaciones que se usarán con el fin de conseguir acceso a recursos protegidos. El usuario debe completar los campos que se observan en la figura 4.18. El significado de estos campos así como su contenido se exponen a continuación:

- *Application Name*:

  El nombre que el usuario elija otorgar a su aplicación. Debe de ser corto y describir si es posible la función de esta aplicación, como por ejemplo "ZoteroUploadApp". Este nombre será el que aparezca en el proceso de autenticación cuando el servidor de autorización de Zotero pregunte al usuario si desea autorizar a esta aplicación a realizar acciones en nombre del usuario.

- *Application Website*:

  Se completará con la URL en la que residirá la aplicación. En este ejemplo:

  "https://localhost:5000/oauth".

- *Application Description*:

  Se completará con una breve descripción de la funcionalidad de la aplicación. Este campo es importante pues permite identificar la aplicación con más facilidad posteriormente, si se dispone de muchas.

- *Application Type*:

  El usuario seleccionará el tipo de aplicación dependiendo del lugar en el que se ejecutará. Las opciones son las presentadas continuación y solo puede ser escogida una de ellas:

  Client, si la aplicación se ejecutará desde el escritorio.

  Browser, si la aplicación se ejecutará desde un buscador web.

- *Callback URL*:

  El usuario escribirá la URL a la que la aplicación rediccionará a un usuario tras una autenticación correcta.

Para saber más sobre la función de estos parámetros, leer el apartado dedicado al flujo de autorización OAuth 1.0 dentro de la sección 3.2.2

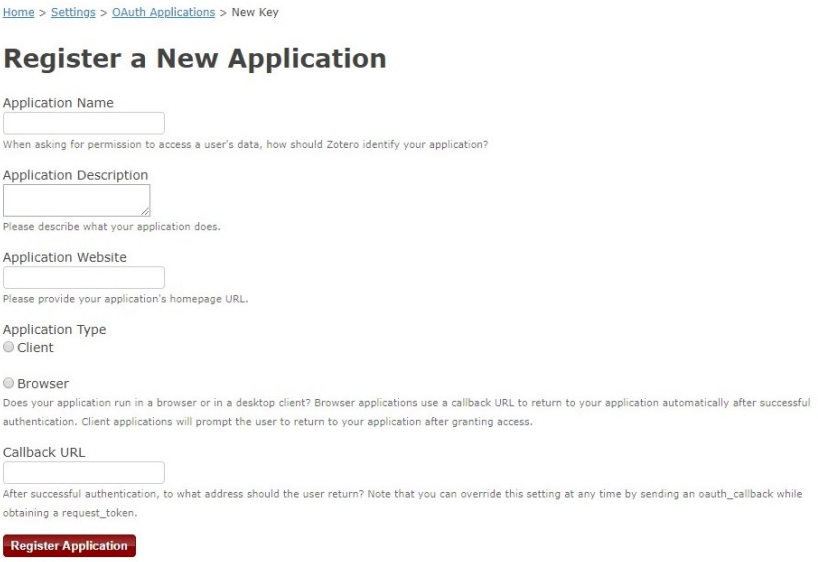

Figura 4.18: Pantalla de registro de aplicación en Zotero

Una vez rellenados estos campos, se pulsará el botón *Register Application*. La página redirigirá automáticamente al usuario de vuelta a la página de la figura 4.17. Esta vez, la página tiene el aspecto de la figura 4.19, en ella puede encontrarse la nueva aplicación creada, y en su descripción se podrá encontrar toda la información necesaria para su uso en las interacciones con la API. Además de los campos introducidos en su creación, también han sido generados los parámetros *Client Key* y *Client Secret*, que se utilizarán más adelante.

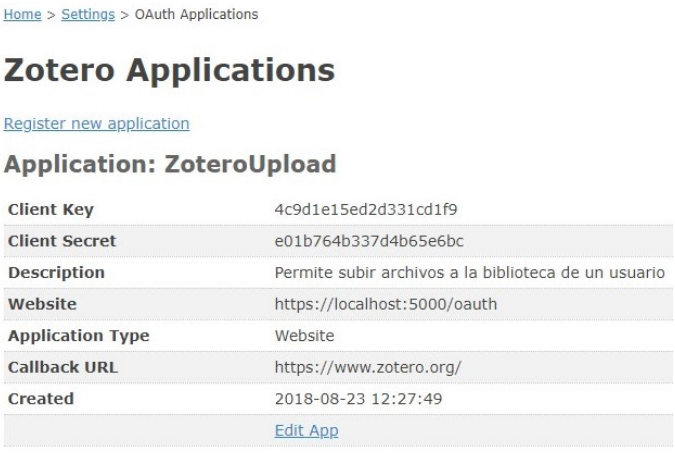

Figura 4.19: Pantalla de información sobre una aplicación en Zotero

Como se menciona previamente, cuando el usuario crea una aplicación, se obtienen los parámetros *Client* y *Client Secret* que serán utilizados en futuras interacciones entre esta aplicación y Zotero, tanto en la autenticación como en los procesos de escritura.

4.2.3. Obtención de un *User ID* y *API key* en Zotero

El tercer paso de este manual se centra en la obtención de credenciales de identificación y escritura. Para poder acceder y escribir datos en la biblioteca de un usuario de Zotero son necesarios estos parámetros:

- *User ID*

  El *User ID* se refiere al identificador de la biblioteca personal del usuario de Zotero. Este identificador es un número positivo de siete dígitos, por ejemplo "5050363". Es la identificación que se usará en las peticiones a la API de Zotero.

- *API key*

  La *API key* de Zotero es la llave que habilita el acceso a recursos protegidos. Cada llave es única y solo sirve para acceder a los datos del usuario con el que esté relacionada, estas llaves no caducan y pueden ser usados tantas veces como sea conveniente. La *API key* deberá ser enviada en cada petición de escritura que se realice a la biblioteca de un usuario de Zotero.

El *User ID* está localizado en la URL "https://www.zotero.org/settings/keys". A diferencia del método de creación de aplicaciones, esta URL puede ser accedida siguiendo el camino desde la URL de inicio de Zotero, *Home*, *Settings*, *Feeds/API*. La página en la cual se encuentra el *User ID*, será similar a la figura 4.20. En ella se observa que el número que se desea obtener se encuentra sobre del botón *Create new private key*. Esta URL es la misma desde donde se administrarán las llaves de acceso.

Figura 4.20: Pantalla de consulta de *User ID* en Zotero

Como se mencionó previamente la URL usada para obtener una *API key* es la misma desde la que se consulta el *User ID* de un usuario. La obtención de una *API key* empieza por hacer clic en *Create new private key*. Cuando el usuario pincha en esta opción, Zotero le dirigirá automáticamente a la página mostrada en la figura 4.21, donde se rellenarán los siguientes campos:

- *Key Description*:

  Una breve descripción de la funcionalidad de la *API key* para identificarla con mayor facilidad posteriormente.

- *Personal Library*:

  El usuario seleccionará los campos a los que quiere permitir acceso a la aplicación que use la *API key*.

  - *Allow library access*: Permite acceso a la aplicación para leer el contenido de la biblioteca del usuario.
  - *Allow notes access*: Permite acceso a la aplicación para leer las notas del usuario.
  - *Allow write access*: Concede permisos de escritura en la biblioteca del usuario a la aplicación. Para este trabajo esta opción es la más importante pues habilitará a la aplicación que use esta llave a subir archivos a la biblioteca de un usuario.

- *Default Group Permissions*:

  Se seleccionarán los permisos de todos los grupos de la aplicación que use la *API key*.

  - *None*: Deniega el acceso de la aplicación a los grupos presentes y futuros.
  - *Read Only*: Permite acceso a la aplicación para leer los grupos presentes y futuros.
  - *Read/Write*: Concede permisos de escritura y lectura a la aplicación para los grupos presentes y futuros.

- *Specific Groups*:

  Seleccionar *Per Group Permissions* permite establecer permisos individuales a cada grupo.

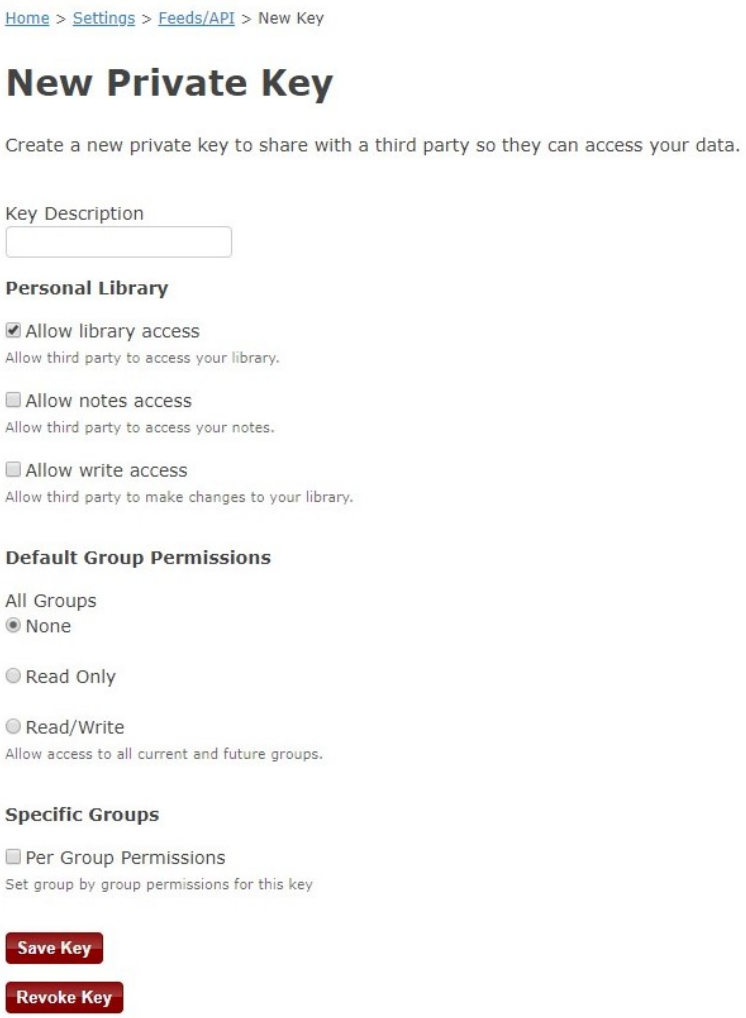

Figura 4.21: Pantalla de registro de una *API key* para Zotero

Una vez estos campos hayan sido rellenados, el usuario pulsará el botón *Save key*. Entonces Zotero le redirigirá a la página de la figura 4.22. En esta página el usuario observará la *API key* que se ha generado según los parámetros introducidos anteriormente. El usuario se hace responsable de que la *API key* sea usada por agentes autorizados. Asimismo, el usuario debe copiar y almacenar este código en un lugar seguro, pues no volverá a ser accesible después de dejar la página actual.

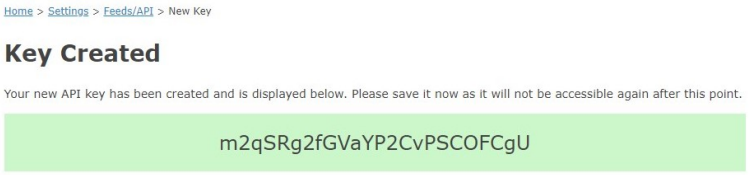

Figura 4.22: Pantalla de registro usuario para Zotero

4.2.4. Preparación del entorno

Hasta ahora toda la información presentada en el manual de uso es independiente del sistema operativo del ordenador (MacOS, Linux, Windows) y el navegador web utilizado (Firefox, Chrome, Internet Explorer). Los siguientes pasos de este manual se realizaron y pueden llevarse a cabo en ordenadores con sistema operativo Windows. Antes de poder ejecutar el programa el usuario debe cumplir varios requisitos.

- Disponer de una plataforma desde la que ejecutar el programa. En este caso es el programa JetBrains PyCharm Community Edition 2018.2 y el lenguaje de programación Python 3.6.

- Descargar y localizar dentro de su ordenador el archivo de la referencia bibliográfica con el que trabaja este manual a la biblioteca de Zotero .

- Asegurarse de que el código que controla el programa se ejecute sin problemas.

Descarga e instalación PyCharm Community Edition y Python 3.6

En este apartado se explicará el proceso de obtención e instalación de PyCharm. El usuario necesita una plataforma desde la cual ejecutar y compilar el código que controla el programa de gestión de referencias. En el caso de este manual el programa usado será JetBrains PyCharm Community Edition 2018.2. Si el usuario no dispone previamente del programa, el primer paso es hacerse con una versión autorizada de este. El software puede ser obtenerse desde la página web de JetBrains en el apartado de PyCharm en la siguiente URL “https://www.jetbrains.com/PyCharm/”. El programa cuenta con dos versiones para su descarga, una versión *Premium* de pago, con un periodo de prueba, y una versión *Open Source* gratuita; esta última será la versión utilizada en este manual. El software será descargado e instalado, siguiendo los pasos presentados por el software de instalación. Durante este proceso el usuario podrá elegir el directorio en el cual se instalará dentro de su ordenador.

Figura 4.23: Logo del programa PyCharm

En caso de que el ordenador en el cual se esté llevando a cabo el presente tutorial no tiene instalado alguna versión del lenguaje de programación Python, el usuario deberá instalarlo. Python puede ser obtenido en la siguiente URL "https://www.python.org/", en el apartado de descargas. Como puede observarse existen varias versiones de este lenguaje. Las más recientes disponen de un mayor número de funcionalidades que las antiguas, pero al llevar menor tiempo publicadas no tienen tanto soporte. Es por esto que algunos usuarios prefieren programar en versiones anteriores, por otro lado las versiones antiguas con el tiempo dejan de ser soportadas por desarrolladores y pueden quedar obsoletas. Para este trabajo se utilizó la versión de Python 3.6. El usuario elegirá la versión más reciente de Python 3 en el momento de llevar a cabo el manual, la descargará e instalará en el ordenador en el cual se ejecute el programa de gestión de referencias.

Figura 4.24: Logo del lenguaje de programación Python

Una vez completada la instalación de los elementos necesarios, el usuario puede continuar con los siguientes pasos.

Obtención de los archivos necesarios para la ejecución del programa

El objetivo de este manual de usuario es subir el archivo de una referencia bibliográfica específica a la biblioteca de un usuario de Zotero. Es por ello que el usuario debe descargar, en primer lugar el código de Python que es la base del programa de gestión y en segundo lugar, el archivo que contiene la referencia bibliográfica. Dichos elementos serán almacenados en un directorio conocido en el ordenador del usuario. Tanto el código del programa como el archivo de la referencia bibliográfica se encuentran disponible en el repositorio de Github [3]. Una vez obtenidos los archivos se puede continuar con el siguiente paso del manual de uso.

Preparación del entorno para la ejecución del programa

En esta sección del manual se creará un archivo Python que contendrá el código encargado de la ejecución del programa. También se asegurará su correcto funcionamiento. Lo primero que hará el usuario es ejecutar el programa PyCharm Community Edition 2018.2. En caso de ser la primera vez que se ejecuta este programa en un ordenador, aparecerá un breve tutorial que contiene conceptos generales sobre su uso. Después de este tutorial se abrirá una ventana como la que se muestra en la figura 4.25. En dicha ventana el usuario elegirá la ubicación del proyecto (rojo) y el nombre de este. De no cambiar la ubicación, por defecto los proyectos se crearán en el directorio donde se instaló el programa PyCharm. Por último, se pulsará el botón *Create* (naranja).

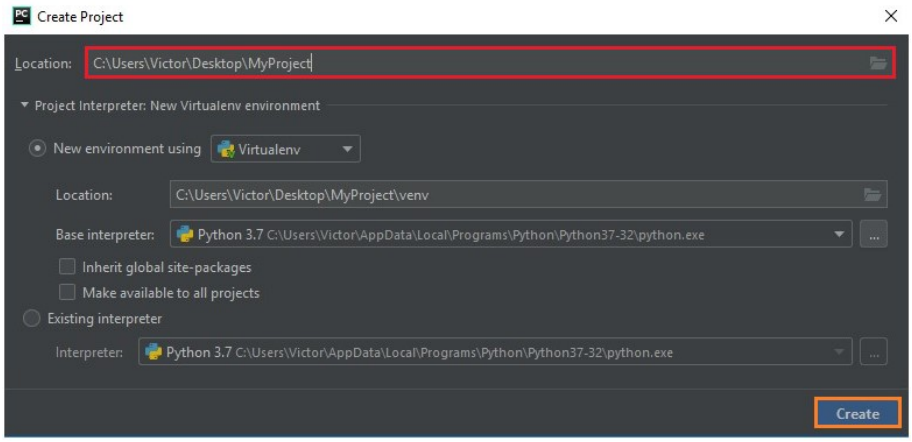

Figura 4.25: Ventana de creación de nuevo proyecto PyCharm

[3] GitHub, "victordiazobregon". [En línea] Disponible en: https://github.com/victordiazobregon/Trabajo-de-fin-de-grado-V-ctor-D-az (último acceso 28 de agosto de 2018)

Una vez que el proyecto se haya generado, el usuario se encontrará en la ventana principal de administración de un proyecto como se puede observar en la figura 4.26 (rojo). El único contenido de dicha ventana es la lista de bibliotecas que se instalan por defecto al crear el proyecto.

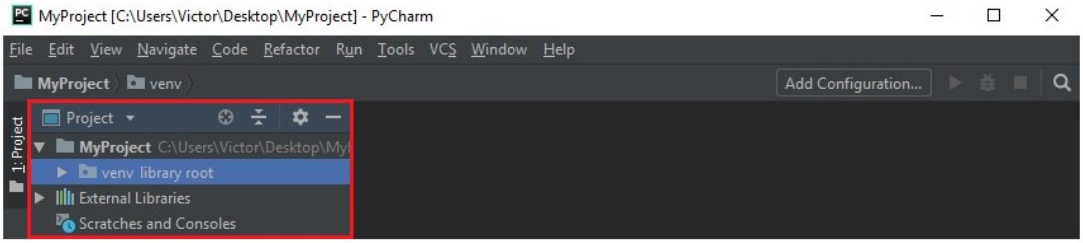

Figura 4.26: Ventana de administración de un nuevo proyecto en PyCharm

Se procederá a crear un archivo Python (ver figura 4.27) en el que añadir el código que se descargó en el paso anterior del manual. Primero, se asegurará de que está seleccionado *MyProject* en la ventana de administración del proyecto y después se pulsará en la pestaña de *File* (rojo). Se abrirá un desplegable en el cual se deberá seleccionar la opción *New* (naranja). Al pulsarla aparecerá una lista de la cual se elegirá la opción *Python File* (verde). Al elegir dicha opción aparecerá una ventana (azul) con el campo *Name* vacío para ser completado. El usuario introducirá aquí el nombre del archivo que se va a crear.

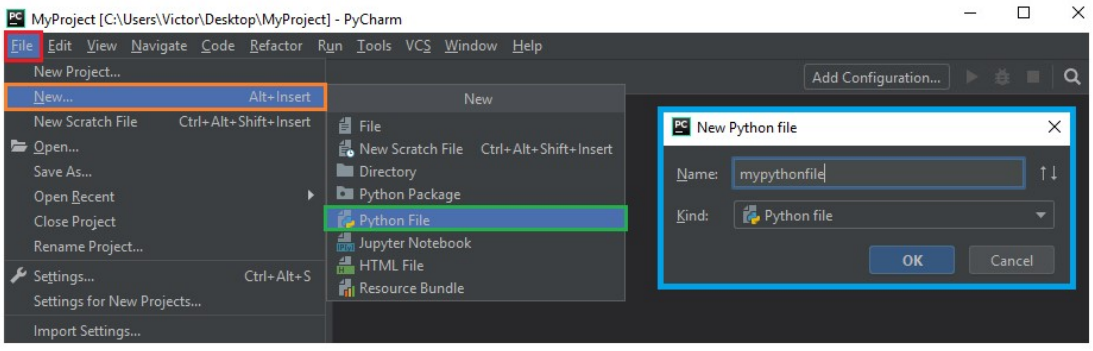

Figura 4.27: Proceso de creación de un archivo Python

Al pulsar el botón *OK* finalizará el proceso de creación del archivo. A partir de este momento el usuario puede observar que se encuentra en la pestaña del archivo creado (ver figura 4.28) y que en la ventana de administración del proyecto (rojo) ha aparecido el archivo de Python creado (amarillo).

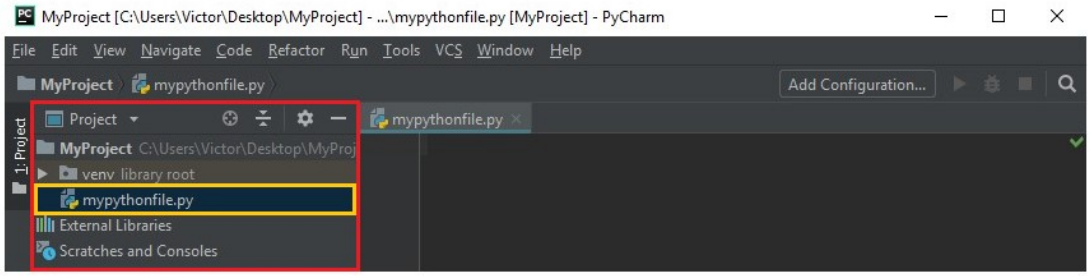

Figura 4.28: Ventana de administración del proyecto tras la creación de archivo Python

Después de que el usuario ha generado el archivo Python puede proceder a añadir el código obtenido en el paso anterior del manual. Antes de poder ejecutar el código se debe revisar en busca de posibles errores. Estos errores pueden estar generados debido a que los *imports* necesarios para el correcto funcionamiento del código no estén instalados (ver figura 4.29). Para instalarlos simplemente habrá que hacer clic en el módulo que no esté instalado; el cual se podrá identificar al estar subrayado de color rojo (rojo). Al hacer clic aparecerá un icono con forma de bombilla sobre el módulo (amarillo) y al hacer clic en este icono, aparecerá la opción *install* seguido del nombre del módulo (verde). Al pulsar en *install* comenzará automáticamente la instalación del módulo. Si no hubo ningún error durante el proceso aparecerá la ventana *Package installed successfully* (azul).

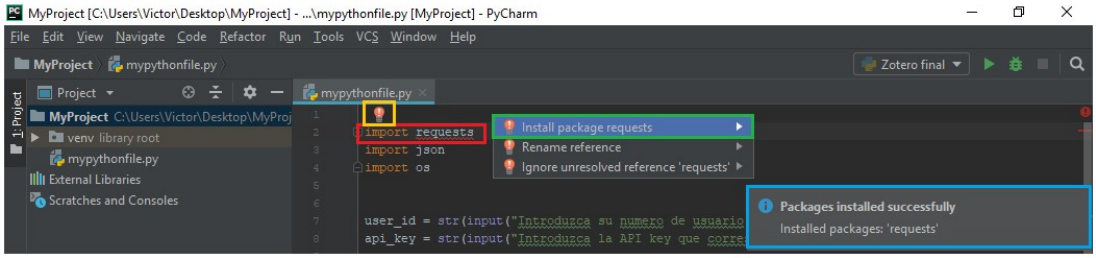

Figura 4.29: Proceso de instalación de un módulo

Otro posible error es la localización del archivo que se subirá a la biblioteca de Zo-

tero. En el caso de este manual el directorio es C:/Users/Victor/Desktop/TFG. Si el archivo utilizando se encuentra en otro directorio, el usuario debe especificarlo. Para ello cambiará el valor de la variable *path* e introducirá el nuevo directorio, es muy importante cambiar el símbolo "/"por "//". Después de comprobar que no existen errores se procederá a ejecutar el código.

4.2.5. Ejecución del gestor de referencias

Tras completar todos los pasos anteriores del tutorial, el usuario puede finalmente ejecutar el código del programa (ver figura 4.30). Para ello se pulsará el botón *Run* (rojo) que abrirá un desplegable en el cual se elegirá la opción *Run* (amarillo). Se abrirá una ventana (azul) que contendrá los archivos que pueden ejecutarse. De dicho desplegable se elegirá el archivo en que fue añadido el código del programa, en este caso, *mypythonfile* (verde).

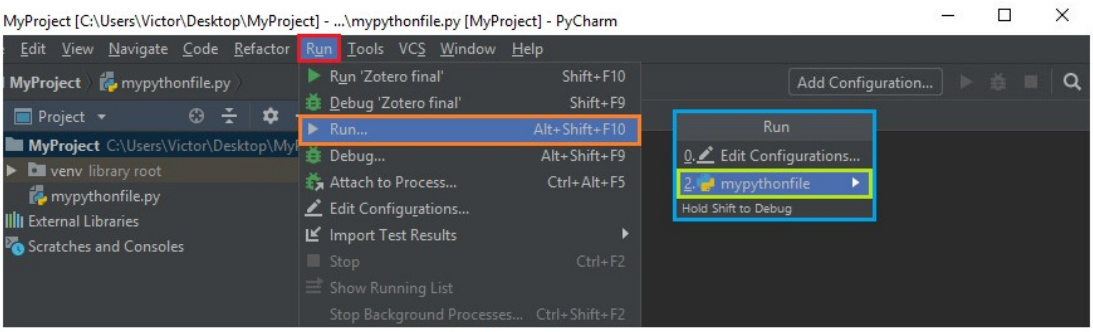

Figura 4.30: Ventana de administración del proyecto tras la creación de archivo en PyCharm

Una vez seleccionada la opción el código comenzará a ejecutarse. En la parte inferior de la ventana del proyecto se encuentra la consola (ver figura 4.31). Dicha consola es utilizada para seguir el avance del código e introducir los datos que el programa requiera. En primer lugar, el programa preguntará al usuario por su *UserID* (rojo) que es el ID de la biblioteca personal del usuario. A continuación, se solicita la *API key* (naranja) que corresponde al *UserID* introducido. Estos parámetros fueron obtenidos por el usuario en la sección 4.2.3 de este manual.

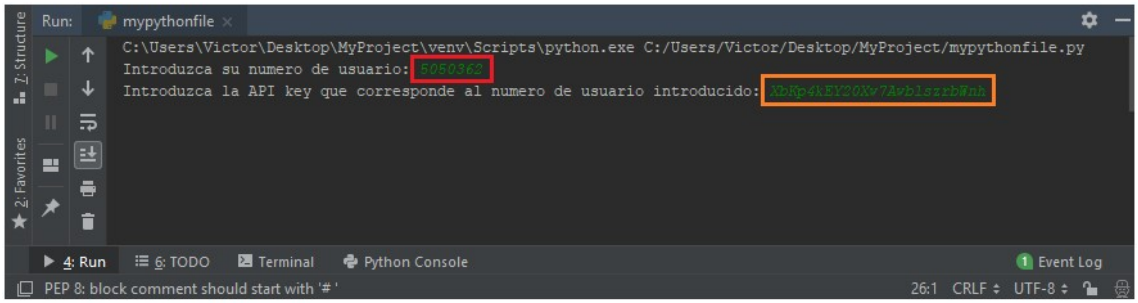

Figura 4.31: Ejecutar un archivo Python en PyCharm

En este tutorial el proceso que permite elegir un documento que contenga una referencia desde el ordenador del usuario no se puede ejecutar, pues ya está definido el archivo que se usará en la demostración. En el apartado 4.1.2 se encuentra una descripción extensa de los procesos que lleva a cabo el programa para finalmente subir el archivo de la referencia a la biblioteca de Zotero.

El programa continúa ejecutándose. En caso de que los datos introducidos fueran válidos, una referencia será subida a la biblioteca de Zotero del usuario. Por el contrario, si los datos introducidos por el usuario son erróneos, el programa presentará el código de error devuelto por el servidor acompañado de una descripción del error experimentado.

Capítulo 5

Conclusiones

Este último capítulo está dedicado al análisis del proceso de investigación y a los resultados obtenidos en este trabajo. Para ello primero se realizará un estudio del proceso de investigación, seguido de un estudio del trabajo de implementación. Finalmente, se puede encontrar una sección dedicada al estudio del futuro del proyecto.

5.1. Conclusiones

En esta sección se hablará de los resultados obtenidos en la realización de este trabajo. Para ello se distinguen dos partes en el TFG, la primera dedicada a la investigación y familiarización con las tecnologías web y gestores de referencias bibliográficas y una segunda parte dedicada a la implementación de estos conceptos para crear un programa capaz de gestionar referencias.

- Investigación

  El trabajo completado forma parte de un proyecto más complejo y con posible relevancia a nivel internacional en el mundo científico. En el planteamiento del trabajo se introduce la motivación que ha impulsado este trabajo y se define una solución, que será estudiada. El objetivo del trabajo es reducir el tiempo que emplean los investigadores en administrar su trabajo en diferentes gestores de referencias. Para ello se quiere desarrollar una aplicación o plataforma que combine a un conjunto de gestores de referencias bibliográficas ya existentes. Con el objetivo de acceder desde una única aplicación a los documentos de varias cuentas como Zotero o ResearchGate, sin necesidad de acceder a cada una de ellas individualmente.

Para conseguir este objetivo, es preciso acceder y compartir datos de usuarios de la red. Es por ello que existe un apartado dedicado exclusivamente al marco legal. En este apartado se trata la protección de datos en Internet y los organismos que la regulan. También es necesario situar las tecnologías web, los gestores de referencias y el propio trabajo, en el entorno socio-económico, para concretar el posible impacto de estos y como benefician a la sociedad.

La integración de los gestores de referencias en el proyecto depende de la naturaleza de estos. Es por ello que en el trabajo se centra en el estudio de una selección de estas plataformas. En la memoria primero se realiza un estudio de los servicios que ofrece cada gestor de cara a sus usuarios, tales como el número de miembros que posee o su presencia en la comunidad científica. En segundo lugar, se lleva a cabo un examen de las herramientas de conectividad de la plataforma, como las funciones de su API y la disponibilidad de esta. Con estos factores se determina la viabilidad de la integración en la aplicación.

Para poder continuar con la implementación es necesario conocer en profundidad las tecnologías en las que se fundamentan las APIs de los gestores estudiados. Encontramos así la sección del estado del arte dedicada a la tecnologías web, en concreto a las APIs, para qué sirven, cómo interactúan entre servicios web, cómo realizar peticiones de información e interpretar respuestas de servidores y a los protocolos de autenticación web OAuth, distinguiendo entre sus dos iteraciones OAuth 1.0 y OAuth 2.0, explicando en detalle el flujo de información en ellas.

- Implementación

Los objetivos de este trabajo se han mantenido constantes a lo largo del tiempo, pero la plataforma de gestión sobre los que aplicarlos ha cambiado varias veces. Estos objetivos son: conseguir de manera automática autenticación para acceder a recursos protegidos de un servicio web y la gestión de referencias sin intervención de un usuario. En el comienzo de el trabajo, la plataforma *ReserachGate* fue la elegida y tras su estudio se determinó que su integración era inviable debido a la ausencia de API pública.

Después de este intento fallido se continuó con el desarrollo en el gestor ORCID. ORCID poseía una API pública y una documentación extensa sobre su uso. El problema surgió al descubrir que tres cuartos de las funciones que proporcionaba su API estaban escondidas detrás de una barrera de pago. Las funciones reservadas para miembros *Premium* no pudieron ser accedidas y se decidió una vez más cambiar de plataforma.

El siguiente gestor elegido fue la popular red social Linkedin que, como ORCID, poseía una API pública, esta vez con todas sus funciones disponibles a todos los usuarios y una extensa documentación. La integración con Linkedin avanzó hasta la obtención de autenticación y se detuvo cuando se descubrió que el acceso a la API había sido ampliamente restringido, necesitando solicitar permisos de desarrollador a Linkedin para poder acceder a datos de usuarios, permisos que podrían llegar a tardar en concederse tres meses.

Finalmente, comenzó la implementación de Zotero. Aunque en esta plataforma se cumplieron los objetivos marcados, en el proceso hubo que hacer compromisos en el diseño. El más importante de estos es la obtención de los permisos para acceder a los datos protegidos. En un principio este proceso debería ser completamente automático, pero se decidió utilizar la herramienta integradas en la página web de Zotero que genera las claves que permiten este acceso.

5.2. Planes futuros

En este apartado se comentarán los planes de futuro de este trabajo de fin de grado. Como ya se había mencionado en el apartado 1.1, el objetivo final consiste en la creación de una aplicación de escritorio que permita gestionar archivos de varios gestores de referencias, desde una misma plataforma. Hasta ahora, un usuario que necesite publicar la misma referencia en los gestores bibliográficos Zotero y ORCID, deberá iniciar sesión con credenciales distintas en cada página y seguir el protocolo de cada página para subir el mismo documento.

Para conseguir las metas de ahorrar recursos y tiempo al usuario se quiere crear esta aplicación. Pero para lograr el objetivo último, se deben completar las siguientes tareas.

- Integración

  Se debe incluir el mayor numero de gestores de referencias bibliográficas dentro de la aplicación final. Este es el mayor obstáculo a sobrepasar en el futuro, pues muchos de los gestores de referencias bibliográficas más importantes del momento como ResearchGate y Google Scholar no disponen de APIs públicas que ayuden a autenticar y gestionar archivos, funciones imprescindibles para el propósito final. Para ello los continuadores de este trabajo deben, o conseguir acceso a estas herramientas o bien desarrollar unas nuevas que habiliten el fin.

- *Graphical User Interface GUI*

  Los gestores de referencias bibliográficas cuya integración haya sido posible serán incluidos en una aplicación que unifique la experiencia de uso de las diferentes plataformas de gestión de referencias. Para ello la aplicación utilizará los códigos individuales de autenticación y gestión de archivos y permitirá a sus usuarios administrar su trabajo, eliminando la necesidad de registrarse en cada interacción con cada gestor. La aplicación ha de tener una interfaz de usuario simple e intuitiva que permita al usuario acceder a sus carpetas y seleccionar los documentos que desee subir e indicar a qué gestor desea subirlos. A continuación, se presenta el concepto de la interfaz de usuario de la aplicación.

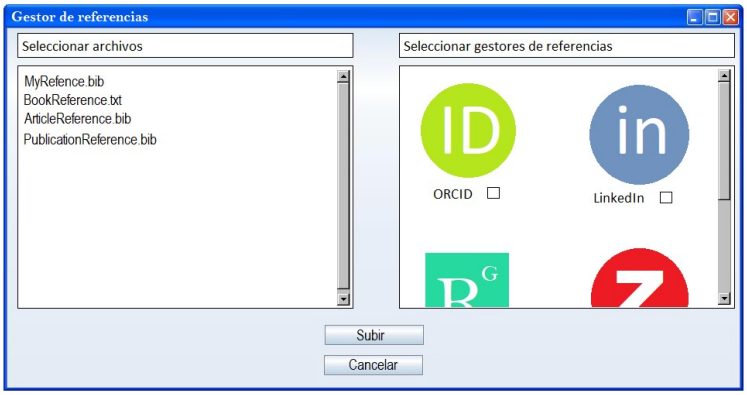

Figura 5.1: Concepto de la interfaz de usuario del gestor de referencias

- Promoción

  Una vez completada la aplicación, sera necesario llevar a cabo un estudio de mercado, seguido de una campaña de marketing, para conseguir que la aplicación tenga la mayor visualización y uso posibles.

Bibliografía

[1] José María Baño León. La distinción entre derecho fundamental y garantía institucional en la constitución española. *Revista española de derecho constitucional, nº 24, pp. 155-179*, 1988.

[2] Juan Carlos I Rey de España. Ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal. *Boletín Del Estado*, vol. 298, nº 2, pp. 43088-43099, 1999.

[3] Roberto Mayor Gómez. Contenido y novedades del reglamento general de protección de datos de la ue (reglamento ue 2016/679, de 27 de abril de 2016). *Gabilex: Revista del Gabinete Jurídico de Castilla-La Mancha, nº 6, pp. 243-280*, 2016.

[4] Carmen Cheliz. El derecho al olvido digital. una exigencia de las nuevas tecnologías, recogida en el futuro reglamento general de protección de datos. 2016.

[5] Flavia Baladán and Jimena Hernández Varela. Intimidad y privacidad frente a la interceptación de las comunicaciones electrónicas. In *XVI Simposio Argentino de Informática y Derecho (SID 2016)-JAIIO 45 (Tres de Febrero, 2016)*, 2016.

[6] Fernando Herrera González. La aplicación de principios de derecho de competencia a la regulación sectorial de telecomunicaciones. *Información comercial española, nº 832*, page 45, 2006.

[7] Registro General de Protección de Datos. Agencia española de protección de datos, 2016.

[8] Ricard Martínez Martínez. El derecho fundamental a la protección de datos: perspectivas. *IDP: revista de Internet, derecho y política= revista d'Internet, dret i política, nº 5, pp. 47-61*, 2007.

[9] Javier Álvarez Hernando. *Guía práctica sobre Protección de Datos: cuestiones y formularios (e-book)*. Lex Nova, 2011.

[10] José Luis Orihuela. Internet: la hora de las redes sociales. *Nueva revista*, vol. 119, pp. 57-62, 2008.

[11] José-Antonio Cordón-García, Helena Martín-Rodero, and Julio Alonso-Arévalo. Gestores de referencias de última generación: análisis comparativo de refworks, endnote web y zotero. *El profesional de la información*, vol. 18, nº 4, pp. 445-454, 2009.

[12] Enrique Orduña-Malea, Alberto Martín-Martín, and Emilio Delgado López-Cózar. Researchgate como fuente de evaluación científica: desvelando sus aplicaciones bibliométricas. *El profesional de la información (EPI)*, vol. 25, nº 2, pp. 303-310, 2016.

[13] K Mueen Ahmed and Bandar Al Dhubaib. Zotero: A bibliographic assistant to researcher. *Journal of Pharmacology and Pharmacotherapeutics*, vol. 2, nº 4:303, 2011.

[14] Tim Berners-Lee, Roy Fielding, and Henrik Frystyk. Hypertext transfer protocol–http/1.0. Technical report, 1996.

[15] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. .º'Reilly Media, Inc.", 2011.

[16] Stefan Tilkov. A brief introduction to rest. *InfoQ, Dec*, vol. 10, 2007.

[17] John Franks, Phillip Hallam-Baker, Jeffrey Hostetler, Scott Lawrence, Paul Leach, Ari Luotonen, and Lawrence Stewart. Http authentication: Basic and digest access authentication. Technical report, 1999.

[18] Ryan Boyd. *Getting started with OAuth 2.0*. .º'Reilly Media, Inc.", 2012.

[19] Dick Hardt. RFC 6794: The OAuth 2.0 authorization framework, IETF. Technical report, 2012.

[20] Michael Jones and Dick Hardt. RFC 6794: The OAuth 2.0 authorization framework: Bearer token usage, IETF. Technical report, 2012.

[21] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[22] Tim Bray. The javascript object notation (json) data interchange format. Technical report, 2017.